
AUDIO EMBEDDINGS FOR SEMI-SUPERVISED
ANOMALOUS SOUND DETECTION

DISSERTATION

zur Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

KEVIN WILKINGHOFF

aus

HAMM

Bonn, 2024

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen
Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: apl. Prof. Dr. Frank Kurth
2. Gutachter: Prof. Dr. Reinhard Klein

Tag der Promotion: 2. September 2024
Erscheinungsjahr: 2024

ABSTRACT

Detecting anomalous sounds is a difficult task: First, audio data is very high-dimensional and anomalous signal components are relatively subtle in relation to the entire acoustic scene. Furthermore, normal and anomalous audio signals are not inherently different because defining these terms strongly depends on the application. Third, usually only normal data is available for training a system because anomalies are rare, diverse, costly to produce and in many cases unknown in advance. Such a setting is called semi-supervised anomaly detection. In domain-shifted conditions or when only very limited training data is available, all of these problems are even more severe.

The goal of this thesis is to overcome these difficulties by teaching an embedding model to learn data representations suitable for semi-supervised anomalous sound detection. More specifically, an anomalous sound detection system is designed such that the resulting representations of the data, called embeddings, fulfill the following desired properties: First, normal and anomalous data should be easy to distinguish, which is usually not the case for audio signals because the definition of anomalies is entirely application-dependent. Second, in contrast to audio signals that are very high-dimensional and may have different durations or sampling rates and thus are difficult to handle, embeddings should have a fixed and relatively low dimension. Third, audio signals may have been recorded under very different acoustic conditions leading to strong variability between signals that, from an anomalous sound detection perspective, is not desired. Ideally, embeddings used for detecting anomalies should be mostly insensitive to these acoustic changes and only sensitive to their degree of abnormality.

The main contributions of this thesis are the following: First and foremost, angular margin losses, namely sub-cluster AdaCos, AdaProj and TACos, specifically designed to train embedding models for anomalous sound detection and for few-shot open-set sound event detection are presented. In various experiments, it is shown that the embeddings obtained with these loss functions outperform embeddings obtained by using other angular margin losses, one-class losses or pre-trained embeddings. As another contribution, it is proven that angular margin losses can be seen as a regularized multi-class version of one-class losses, which helps to cope with background noise. Furthermore, design choices for learning embeddings that are robust to acoustic domain shifts by generalizing well to previously unseen domains are presented, which results in an anomalous sound detection system significantly outperforming other state-of-the-art systems. As a last contribution, it is investigated how to obtain good decision thresholds and a novel performance metric, called F_1 -EV, that measures the difficulty of estimating a good threshold is presented.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor Frank Kurth for his guidance and support throughout the entire time it took me to write this thesis while granting me academic freedom in choosing the research topics I wanted to pursue. Likewise, I would like to thank Reinhard Klein for being willing to serve as a second supervisor as well as Thomas Schultz and Ulrich Schade for participating in the examination and defense of this thesis.

I am equally thankful to my colleagues at Fraunhofer FKIE with whom I had the pleasure to work and collaborate: Paul M. Baggenstoss, Alessia Cornaggia-Urrigshardt, Fabian Fritz, Fahrettin Göçköz, Lukas Henneke, Hans-Christian Schmitz, Sebastian Urrigshardt and all others not mentioned here but who also had their contributions if only by making lunch breaks much more enjoyable.

Furthermore, I would like to thank all members of the DCASE community for discussing ideas and working together on similar research topics. In particular, I would like to mention Yacine Bel-Hadj and Keisuke Imoto for being interested in collaborating with me, Toni Heittola, Annamaria Mesaros and Romain Serizel for including me in organizational duties, and all organizers of the annual anomalous sound detection task for providing most of the datasets used in this thesis.

Last but not least, I am deeply grateful for the continuous support of my wife Kristin, my parents André and Sabine, my siblings Dennis and Melina as well as their other halves Pia and Fabian, my wife's side of the family Ria, Uwe, Heike, Tobias, Angi, Smilla, Lunis, Carlotta, and all of my friends whom I will not try to list here knowing that I will most likely forget to mention too many of them.

CONTENTS

1	INTRODUCTION	1
1.1	Semi-supervised anomaly detection	2
1.2	Audio embeddings	4
1.3	Example applications	5
1.4	Main contributions	7
1.5	Thesis outline	8
1.6	Notation and Preliminaries	8
2	AUDIO EMBEDDINGS FOR ANOMALY DETECTION	9
2.1	Contributions of the author	10
2.2	Input feature representations	10
2.2.1	Pre-processing audio signals	11
2.2.2	Spectral features	12
2.2.3	Normalization	13
2.3	One-class embeddings	13
2.3.1	Autoencoders	14
2.3.2	Compactness loss	16
2.4	Auxiliary task embeddings	19
2.4.1	Angular margin losses	19
2.4.2	Handling imbalanced data	24
2.5	Pre-trained embeddings	24
2.6	Computing an anomaly score	26
2.7	Data augmentation	27
2.7.1	Mixup	27
2.7.2	SpecAugment	28
2.7.3	Simulating anomalies	29
2.8	Comparison of different embedding types	30
2.9	Ensembling	31
2.10	Evaluation metrics	32
2.10.1	Anomaly detection	33
2.10.2	Open-set classification	36
2.10.3	Sound event detection	37
2.11	Decision threshold estimation	39
2.12	Summary	42
3	ANOMALOUS SOUND DETECTION SYSTEM DESIGN	45
3.1	Contributions of the author	45
3.2	Example application: Machine condition monitoring	45
3.2.1	Experimental setup	46
3.2.2	Baseline model for extracting embeddings	46

3.3	Relation between one-class and angular margin losses	48
3.3.1	Compactness loss on the unit sphere	48
3.3.2	Relation between the compactness loss and AdaCos	50
3.3.3	Performance evaluation	51
3.4	Sub-cluster AdaCos	53
3.4.1	Definition	53
3.4.2	Relation to the compactness loss	54
3.4.3	Comparison of backends	58
3.4.4	Utilizing mixup	60
3.4.5	Determining the number of sub-clusters	61
3.4.6	Replacing embeddings with input data statistics	61
3.4.7	Comparison to other published systems	64
3.5	Summary	64
4	DECISION THRESHOLD ESTIMATION	67
4.1	Contributions of the author	67
4.2	Estimating a decision threshold	68
4.2.1	Performance comparison of different estimation methods	68
4.2.2	Choosing a set of observed anomaly scores	70
4.3	F_1 -EV score	72
4.3.1	Definition	72
4.3.2	Experimental setup	75
4.3.3	Experimental comparison with existing evaluation metrics	76
4.3.4	Choosing the hyperparameter $\beta_{F_1\text{-EV}}$	77
4.4	Summary	78
5	DOMAIN ADAPTATION AND GENERALIZATION	81
5.1	Contributions of the author	83
5.2	Machine condition monitoring in domain-shifted conditions	84
5.2.1	DCASE2022 ASD dataset	84
5.2.2	DCASE2023 ASD dataset	85
5.3	Designing an ASD system for domain generalization	86
5.3.1	System description	86
5.3.2	Experimental investigations of individual design choices	89
5.4	Explaining the decisions	92
5.4.1	Visualizing the input as viewed by the model	94
5.4.2	Visualizing the embedding space	96
5.5	Comparison to pre-trained embeddings	98
5.5.1	System design for pre-trained embeddings	98
5.5.2	Experimental results	99
5.6	AdaProj	101
5.6.1	Definition	101
5.6.2	Choosing a sub-space dimension	103
5.6.3	Performance evaluation	104

5.7	Self-supervised learning	104
5.7.1	Approaches	106
5.7.2	Combining multiple approaches	108
5.7.3	Performance evaluation	109
5.8	Putting it all together	111
5.9	Summary	112
6	FEW-SHOT OPEN-SET CLASSIFICATION	115
6.1	Contributions of the author	115
6.2	Few-shot open-set classification	116
6.2.1	Dataset	118
6.2.2	System design	118
6.2.3	Experimental results	119
6.3	Sound event detection application: Keyword spotting	120
6.3.1	Related work	121
6.3.2	Dataset	122
6.3.3	System overview	122
6.3.4	Extracting embeddings	123
6.3.5	TACos	124
6.3.6	DTW backend	128
6.3.7	Baseline systems	128
6.3.8	Experimental comparison	129
6.4	Summary	130
7	CONCLUSION	131
7.1	Summary	131
7.2	Outlook and future work	133
A	APPENDIX	135
A.1	Key publications	135
A.1.1	Key publication 1	135
A.1.2	Key publication 2	144
A.1.3	Key publication 3	150
A.1.4	Key publication 4	163
A.1.5	Key publication 5	169
A.1.6	Key publication 6	175
A.1.7	Key publication 7	183
A.1.8	Key publication 8	189
A.1.9	Key publication 9	195
A.1.10	Key publication 10	201
	LIST OF FIGURES	217
	LIST OF TABLES	219

LIST OF ACRONYMS	221
LIST OF SYMBOLS	224
BIBLIOGRAPHY	229

INTRODUCTION

In general, anomaly detection is the process of distinguishing *normal* data, which are the result of measurements or sensor input, from *anomalous* data, which substantially deviates from normal data in some way. Other even less formal definitions are to see anomalous data as every data sample that strikes the eye or is deemed interesting or perhaps surprising. There are several difficulties to overcome when trying to formalize these vague definitions. First, one needs to define *in which way* and *to what extent* data needs to deviate from normal data in order to be considered anomalous. One can distinguish *weak* anomalies, which are essentially noisy normal samples, and *strong* anomalies that should always be recognized as anomalies [3]. Thus, weak anomalies can be considered normal or anomalous, depending on the application. Furthermore, the term *normal* needs to be precisely defined since it is highly application-dependent and can be influenced by several parameters such as the time and location at which data is collected. Data that is considered normal for a given application can possibly be considered anomalous in another application or even for the same application if one of the data-influencing parameters is changed. In conclusion, one needs a well-defined goal when setting up an anomaly detection system. Sufficient amounts of data need to be collected in the right conditions for training the system such that the underlying distribution of normal data is represented sufficiently well for the application in mind. Otherwise, it cannot be ensured that anomalies detected by the system are truly the ones that should be detected.

Let \mathbf{X} denote an infinite data space such as the space containing all audio signals. Let $\mathbf{X}_{\text{normal}} \subset \mathbf{X}$ and $\mathbf{X}_{\text{anomalous}} \subset \mathbf{X}$ denote the sets of normal and anomalous samples, respectively. For the sake of simplicity, set $\mathbf{X}_{\text{anomalous}} := \mathbf{X} \setminus \mathbf{X}_{\text{normal}}$ as this only requires the choice of normal samples to define both sets. Although using this definition, i.e. using the complement of the normal samples in the set of all theoretically possible samples, $\mathbf{X}_{\text{anomalous}}$ may include samples that never occur in practice, these samples should certainly not be considered normal. The goal of training most anomaly detection systems is to obtain a function $\text{score} : \mathbf{X} \rightarrow \mathbb{R}, \mathbf{x} \mapsto \text{score}(\mathbf{x})$ mapping a data sample \mathbf{x} to an anomaly score denoted by $\text{score}(\mathbf{x})$ such that all normal samples can be separated from anomalous samples. This means that there is a decision threshold $\theta \in \mathbb{R}$ such that

$$\text{score}(\mathbf{x}_n) \leq \theta < \text{score}(\mathbf{x}_a)$$

for all normal samples $\mathbf{x}_n \in \mathbf{X}_{\text{normal}}$ and for all anomalous samples $\mathbf{x}_a \in \mathbf{X}_{\text{anomalous}}$. In other words, anomalous samples should be mapped to much higher anomaly scores than normal samples. It is important to mention that obtaining this function score is only a desired goal and in practice anomalous samples can only rarely be

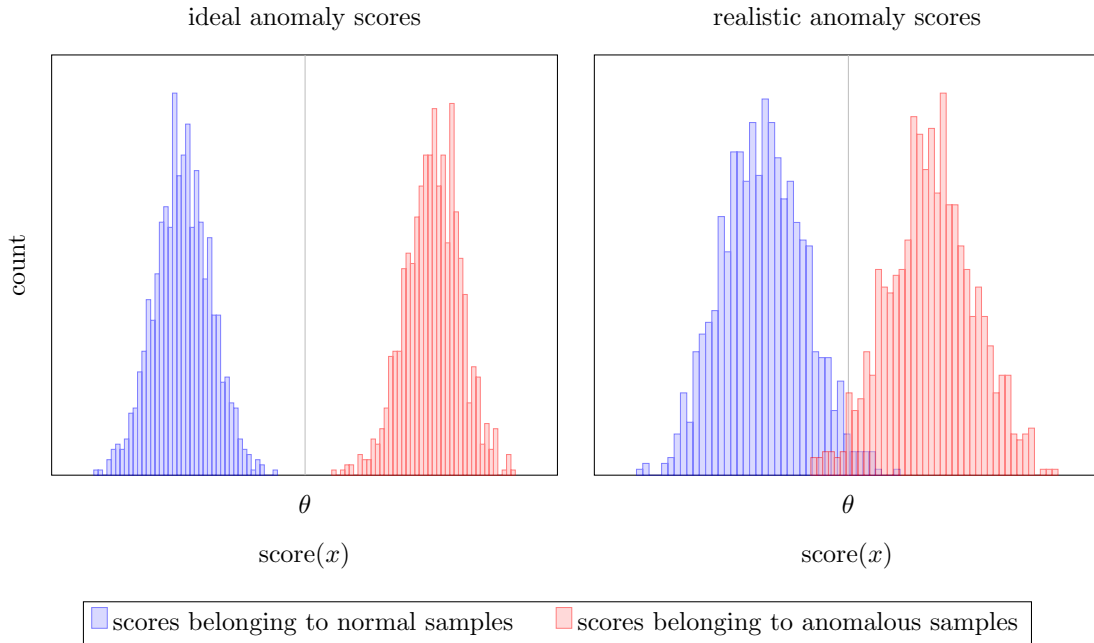


Figure 1: Histograms of anomaly scores and corresponding decision thresholds. On the left, the decision threshold θ perfectly separates the anomaly scores belonging to the normal and anomalous samples. On the right, a perfect separation is not possible because the histograms of the normal and anomalous scores overlap.

perfectly detected, i.e. there is an overlap between the anomaly scores of both sets. This is illustrated in Figure 1.

1.1 SEMI-SUPERVISED ANOMALY DETECTION

Depending on the structure of the training dataset and the given labels for this dataset, there are three different settings in which anomaly detection can take place: *Supervised*, *semi-supervised* and *unsupervised* anomaly detection [3]. These settings are presented in Table 1 and visualized with examples in Figure 2.

Table 1: Overview of different anomaly detection settings.

	training dataset	training labels	data collection
supervised	$X_{\text{train}} \subset X_{\text{normal}} \cup X_{\text{anomalous}}$	available (2 classes)	difficult
semi-supervised	$X_{\text{train}} \subset X_{\text{normal}}$	available (1 class)	moderately difficult
unsupervised	$X_{\text{train}} \subset X_{\text{normal}} \cup X_{\text{anomalous}}$	not available	simple

For *supervised* anomaly detection, the labeled training dataset consists of normal and anomalous data. In non-trivial applications it is impossible to collect all variations of data that are considered anomalous to fully capture the space

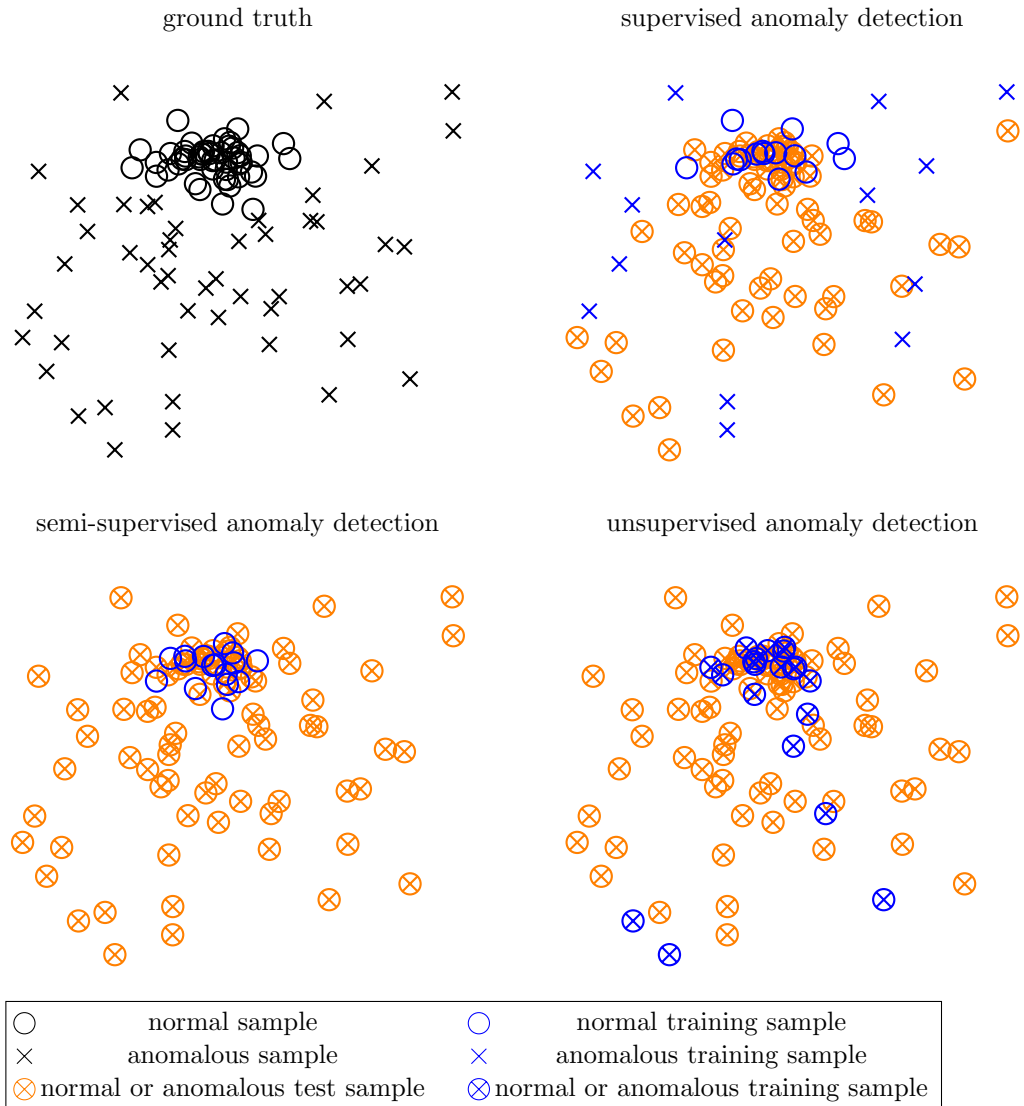


Figure 2: Illustration of different anomaly detection settings. The same dataset is shown with different samples and corresponding labels available for training an anomaly detection system.

of anomalous samples. Thus, only a limited number of boundaries between normal and anomalous samples can be learned. This is especially true in case of high-dimensional data. Still, providing examples of anomalous samples that are of particular interest and thus should always be recognized as being anomalous is useful for training the system. By definition, anomalies only rarely occur. Therefore, it is much more difficult and thus more costly to obtain realistic anomalous samples for training a system than to collect normal data. Furthermore, determining whether a sample is normal or anomalous often requires expert knowledge and can be a cumbersome task. This results in highly imbalanced classes, which poses a problem that needs to be handled appropriately.

In a *semi-supervised* setting, the training dataset contains only normal data¹. For most applications, this is a more realistic setting because it substantially simplifies the data collection process as it only needs to be ensured that all collected samples are in fact normal. However, using only normal data for training leads to worse performance than using a dataset collected in a supervised setting because no a priori knowledge about the expected anomalous data is available.

In some cases it is even impossible to ensure that only normal data is collected and, without explicitly knowing, the training dataset may contain a mixture of normal and possibly anomalous data. This is referred to as an *unsupervised* anomaly detection setting. Compared to both other settings, it is far easier and thus less costly to collect data in this setting. Furthermore, using such a dataset for training a system usually leads to significantly worse performance as only noisy information about the underlying distribution of the normal data is available. In most cases, at least an estimate of the corruption of the training data, i.e. the ratio between normal and anomalous training samples, is needed to obtain useful results.

1.2 AUDIO EMBEDDINGS

Identifying anomalies by analyzing raw audio signals is difficult because of three reasons: First, there is no inherent property separating normal from anomalous samples since defining the normal (and the anomalous) subset is entirely application-dependent. Second, audio signals live in a high-dimensional space with a dimension equal to the duration in seconds times the sampling rate in Hertz and thus contain much redundant information for a given task. Moreover, an audio recording may also consist of several recording channels, which leads to an even higher dimension. Third, individual audio signals may have a different duration, a different sampling rate or may have been recorded in different acoustic conditions or using different sensors and thus a comparison between signals is difficult. To overcome these difficulties for a particular anomalous sound detection ([ASD](#)) application, it would be very favorable if all audio signals were mapped to the same, relatively low-dimensional vector space specifically structured such that representations of normal samples are similar to each other while substantially differing from anomalous samples and also being robust to acoustic variations. In the context of machine learning, such vector representations of the input data are called embeddings. To formalize this, an *ideal embedding function* will be defined as a

¹ If only anomalous data is available for training the system, this is also called a semi-supervised setting. However, for most applications this is rarely the case since it is highly unusual that normal samples are not available and anomalous samples are usually much more difficult to collect than normal samples. Within this thesis the term is reserved for a setting with a training dataset consisting of normal samples only.

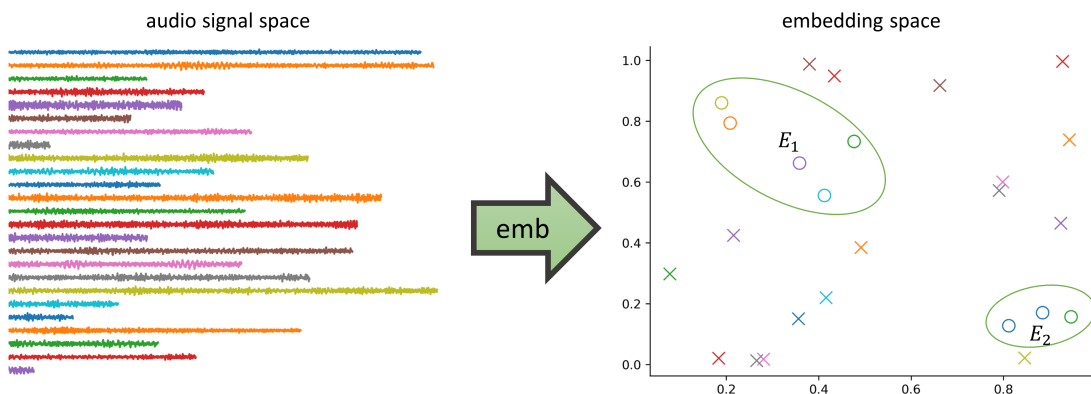


Figure 3: Illustration of an ideal embedding function for ASD.

mapping $\text{emb} : X \rightarrow \mathbb{R}^D$ with $D \in \mathbb{N}$ sufficiently small such that there are $N_e \in \mathbb{N}$ subsets $E_i \subset \mathbb{R}^D$ of the embedding space for which

$$\text{emb}(X_{\text{normal}}) = \bigcup_{i=1}^{N_e} E_i \text{ with } E_i \cap \text{emb}(X_{\text{anomalous}}) = \emptyset \text{ for all } i = 1, \dots, N_e.$$

Such an ideal embedding function is depicted in Figure 3. Note that for such an ideal embedding function it holds that

$$\text{emb}(X_{\text{normal}}) \cap \text{emb}(X_{\text{anomalous}}) = \emptyset.$$

Hence, normal and anomalous samples can be perfectly separated and one can define an anomaly score function that yields perfect results. This is the reason why emb is called an *ideal* embedding function.

Learning an ideal embedding function can be accomplished by using universal function approximators such as neural networks. One of the main difficulties of semi-supervised ASD is the definition of a suitable loss function for training the network because only normal data is available for training. Obtaining an ideal embedding function, i.e. developing a neural network for extracting application-dependent audio embeddings, and designing an ASD system using these embeddings is the goal of this thesis.

1.3 EXAMPLE APPLICATIONS

In [3], the following general applications for anomaly detection are listed: Intrusion detection, credit-card fraud, interesting sensor events, medical diagnosis, law enforcement and earth science. Since the focus of this thesis lies on anomaly detection for audio data, concrete applications will only be given for systems using acoustic data. Numerous applications using other data types will be omitted for the sake of limiting the length of this section. Note that for many applications, an acoustic sensor is only one of multiple different sensors and combining all available information usually leads to much better performance. For example, one can

often utilize video cameras alongside acoustic sensors. Some of the following audio-specific applications coincide with the aforementioned general applications:

- Machine condition monitoring for predictive maintenance [44, 46, 96, 108]: Here, normal sounds are recordings of fully-functioning machines in noisy factory environments and anomalies correspond to mechanical failure. Research for ASD is largely promoted through a machine condition monitoring task of the annual DCASE challenge and will serve as the main example application throughout this thesis.
- Intrusion detection in smart home environments to detect burglary [279]: Here, anomalies are sounds of persons walking through the room and looking for hidden objects in boxes and drawers. Normal sounds are emitted by furniture or electronic devices, people or objects outside or in neighboring rooms and other external sources such as traffic or thunderstorms.
- Medical diagnosis: Normal sounds correspond to a healthy state and any anomaly indicates a disease or other medical conditions. Concrete examples are detecting COVID-19 from speech [160] or detecting heart defects from heart sounds [40].
- Detecting crimes or terrorist attacks with surveillance systems in public places [75, 216]: Here, anomalous sounds consist of gunshots, shattering glass or screams whereas normal sounds are a diverse mixture of sounds such as talking people or passing cars.
- Detecting accidents with road surveillance systems [53, 127]: Normal sounds consist of regular traffic noise and anomalous sounds are mostly crashing cars.
- Detecting interesting events in bioacoustic monitoring as for example novel species or individuals that are seen as anomalies [170] or monitoring the condition of beehives [23] similarly to machine condition monitoring
- Detecting erroneous sensor measurements: This is especially important for wireless sensor networks, e.g. in underwater sensor networks [47] where there can also be issues when collecting and transmitting information.
- Acoustic *open-set classification (OSC)* problems: In addition to a fixed set of known classes that should be recognized, a test sample can also belong to none of these known classes. Therefore, OSC tasks can be decomposed into the sub-tasks anomaly detection and *closed-set classification (CSC)*: Anomaly detection is used to decide whether a given sample belongs to one of the known classes, and thus is considered normal, or to an unknown class, i.e. is an anomalous sample. CSC is used to predict to which of the known classes the sample belongs to. One can also view anomaly detection as a special case

of [OSC](#) with only a single known class denoting all normal samples. Concrete examples are open-set speaker recognition [205] or open-set acoustic scene classification [149]. If not only the class but also the temporal position of a sound event contained in an audio signal of possibly long duration needs to be recognized, this is referred to as sound event detection ([SED](#)) [229, 230]. Typical examples are keyword spotting ([KWS](#)) [136] or bioacoustic event detection [153, 169]. Note that for these applications the a priori likelihood that a normal event occurs is usually much smaller than the likelihood that an uninteresting, i.e. unknown, event occurs.

It is important to emphasize that the goal of all these tasks is not to distinguish specific anomalous sounds from silence or analyze isolated sound events, which would be relatively simple. The acoustic scenes captured by most recordings contain a complex mixture of many different normal sounds, all of which can possibly result in false alarms in case the [ASD](#) system is not performing sufficiently well. Handling this mixture of sounds appropriately is one of the major challenges to overcome in [ASD](#).

1.4 MAIN CONTRIBUTIONS

The main contributions of this thesis are the following:

- Sub-cluster AdaCos, an angular margin loss for semi-supervised [ASD](#), is presented in [Chapter 3](#).
- The relation between one-class losses and angular margin losses is investigated in [Chapter 3](#).
- Multiple methods for obtaining decision thresholds are compared to each other in [Chapter 4](#). Furthermore, F_1 -[EV](#), a novel performance measure taking the estimation of decision thresholds into account, is proposed in the same chapter.
- Several different design choices for making a system robust to (acoustic) domain shifts are proposed and compared to each other in [Chapter 5](#). In particular, the angular margin loss AdaProj, which generalizes the sub-cluster AdaCos loss, and a self-supervised learning framework are presented.
- [TACos](#), a loss function for obtaining embeddings that capture the temporal structure of sound events, is presented and evaluated for few-shot [KWS](#) in [Chapter 6](#).

At the beginning of each chapter, additional contributions are stated.

1.5 THESIS OUTLINE

The remaining parts of this thesis are structured as follows:

- [Chapter 2](#) reviews the state-of-the-art techniques for extracting audio embeddings for semi-supervised [ASD](#) as well as methods for designing [ASD](#) systems based on these embeddings.
- [Chapter 3](#) studies the design of a semi-supervised [ASD](#) system based on audio embeddings.
- [Chapter 4](#) examines how to estimate good decision thresholds that separate the anomaly scores of normal and anomalous samples using only anomaly scores belonging to normal samples.
- [Chapter 5](#) investigates how to design and train an [ASD](#) system such that it is robust against acoustic domain shifts.
- [Chapter 6](#) describes the application of [ASD](#) embeddings to acoustic [OSC](#) and [SED](#) problems.
- [Chapter 7](#) concludes the thesis by summarizing the results and presenting possible directions for future work.

1.6 NOTATION AND PRELIMINARIES

Apart from the notation introduced in the preceding sections, the following notation will be used throughout the thesis:

- $\mathcal{P}(X)$ denotes the power set of X .
- For $N_{\text{classes}} \in \mathbb{N}$ pre-defined classes, the class index function is denoted as $\text{class} : X \rightarrow \{1, \dots, N_{\text{classes}}\}$. The categorical class label function is denoted as $\text{lab} : X \rightarrow [0, 1]^{N_{\text{classes}}}$ with $\sum_{j=1}^{N_{\text{classes}}} \text{lab}(x)_j = 1$ for all $x \in X$. Note that in contrast to the class function, this definition explicitly allows that a sample (partially) belongs to more than a single class.
- Φ denotes the space of permissible neural network architectures for obtaining embeddings and W denotes the corresponding parameter space of the neural networks, i.e. $\Phi = \{\phi | \phi : X \times W \rightarrow \mathbb{R}^D\}$. Unless stated otherwise, specific building blocks of these networks may consist of fully-connected layers, convolutional layers, recurrent layers or any other layers and will only be specified if needed to ensure that all presented results are as general as possible.

All neural networks used for the experimental evaluations of this work are implemented using Tensorflow [1].

The motivation for and the abstract idea of obtaining audio embeddings for semi-supervised [ASD](#) has already been presented in [Section 1.2](#). This chapter provides an overview over different audio embeddings and the state-of-the-art methods for designing an [ASD](#) system with them. An overview of all necessary steps is shown in [Figure 4](#). In general, an [ASD](#) system based on audio embeddings consists of a frontend for extracting input feature representations, a neural network for computing embeddings and a backend for deciding whether an embedding is normal or anomalous. The same structure is also used for machine condition monitoring, which serves as the main example application investigated in this thesis.

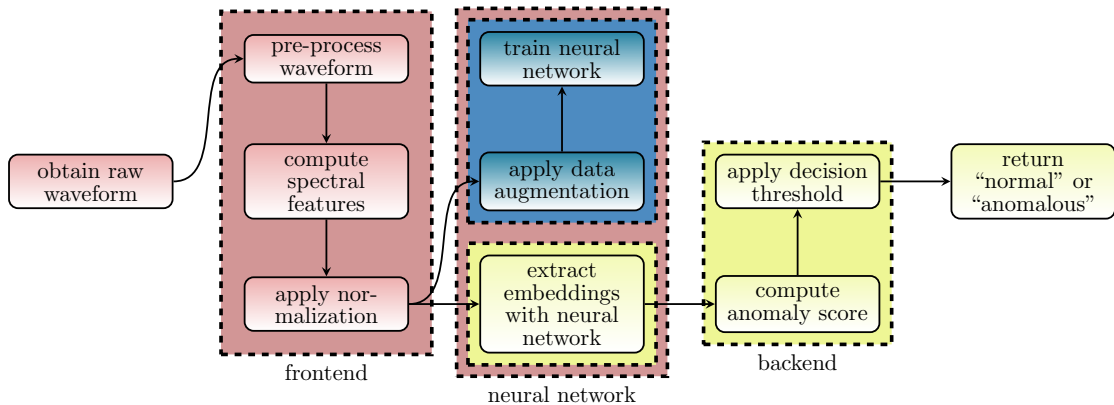


Figure 4: Essential building blocks of a general [ASD](#) system based on audio embeddings. Blocks colored in blue are only used for training the system, blocks colored in yellow are only used for inference and blocks colored in red are used for training and inference.

This chapter is structured as follows: [Section 2.2](#) includes all steps needed for computing input feature representations of the audio signals. These steps are pre-processing the audio signal, calculating spectral features and, optionally, normalizing the features. In [Section 2.3](#), different one-class embeddings and the loss functions for obtaining them are presented as a first type of audio embeddings for [ASD](#). A second type of audio embeddings based on using an auxiliary classification task is reviewed in [Section 2.4](#). [Section 2.5](#) contains a description of a few selected audio embeddings pre-trained on large datasets. In [Section 2.6](#), approaches for calculating an anomaly score are presented. Data augmentation techniques for training the embedding extractor are discussed in [Section 2.7](#). [Section 2.8](#) compares the previously presented three embedding types and [Section 2.9](#) contains approaches for combining several [ASD](#) models. Evaluation metrics for [ASD](#), [OSC](#) and [SED](#)

are presented in [Section 2.10](#). In [Section 2.11](#), methods for estimating a decision threshold are reviewed. The chapter is concluded with a summary in [Section 2.12](#).

2.1 CONTRIBUTIONS OF THE AUTHOR

Some content of this chapter is presented similarly in the following publications:

- Kevin Wilkinghoff and Alessia Cornaggia-Urrigshardt. “On choosing decision thresholds for anomalous sound detection in machine condition monitoring.” In: *24th International Congress on Acoustics*. The Acoustical Society of Korea, 2022.
- Kevin Wilkinghoff and Fabian Fritz. “On Using Pre-Trained Embeddings for Detecting Anomalous Sounds with Limited Training Data.” In: *31st European Signal Processing Conference*. IEEE, 2023, pp. 186–190. DOI: [10.23919/EUSIPCO58844.2023.10290003](#).
- Kevin Wilkinghoff and Frank Kurth. “Why do Angular Margin Losses work well for Semi-Supervised Anomalous Sound Detection?” In: *IEEE/ACM Transactions on Audio, Speech and Language Processing* 32 (2024), pp. 608–622. DOI: [10.1109/TASLP.2023.3337153](#).
- Kevin Wilkinghoff and Keisuke Imoto. “F1-EV Score: Measuring the Likelihood of Estimating a Good Decision Threshold for Semi-Supervised Anomaly Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 256–260. DOI: [10.1109/ICASSP48485.2024.10446011](#).

For publications that are not single-authored, individual contributions of the thesis author and all co-authors to these publications are stated in [Section A.1](#). If not stated otherwise, the content listed in the following paragraph is the sole contribution of the thesis author.

The approaches for estimating a decision threshold as presented in [Section 2.11](#) are also discussed in [\[252\]](#). Alessia Cornaggia-Urrigshardt assisted with reviewing the literature containing these approaches. The same pre-trained embeddings as described in [Section 2.5](#) are also listed in [\[258\]](#). [\[262\]](#) contains a similar definition of the compactness loss ([Definition 2.3](#)) and the angular margin losses ArcFace ([Definition 2.7](#)) and AdaCos ([Definition 2.8](#)). The definition and discussion of AUC-ROC contained in [Section 2.10](#) is based on [\[260\]](#).

2.2 INPUT FEATURE REPRESENTATIONS

Before using audio data as input for neural networks that are trained to extract embeddings, a few pre- and post-processing steps are or may be necessary.

2.2.1 *Pre-processing audio signals*

In many monitoring applications, all recorded audio signals have the same length. The reason is that for a continuous observation it is necessary to specify a recording protocol because anomalous sounds may appear arbitrarily in time. For example, when monitoring the condition of machines in factories, these machines are running without interruption. Unless the process of recording shall continue infinitely, one is forced to decide for a finite recording duration. This duration is application-dependent and should be long enough to ensure that normal and anomalous samples can be distinguished but as short as possible to reduce the total inference time, which also includes the recording time. Furthermore, repeatedly recording for the same fixed duration is to be preferred because this simplifies the recording process and all following steps.

In case a uniform recording length cannot be guaranteed, there are several strategies that can be used to handle recordings with varying lengths. The simplest solution is to unify the duration of all recorded audio signals. Audio signals that are too short are padded with silence or repeated and cut appropriately until the desired length is reached. Audio signals that are too long are cut into overlapping segments of the desired length and the mean of the output of the neural network obtained from these segments can be used as the result for the entire initial recording. Another strategy is to use neural network architectures that can handle varying lengths. One possibility is to use several convolutional layers followed by global pooling layers [128] applied to the temporal axis [228]. A popular example is the x-vector architecture developed for speaker recognition [210]. This architecture uses statistics such as the temporal mean and the temporal standard deviation and stacks them into a single representation as the last layer. Another approach is to use recurrent neural networks such as gated recurrent units [28] or long short-term memories [83] that are capable of handling data with varying time dimension by design. More complex approaches are more powerful but require more computational resources for training as well as more training data to avoid overfitting. Therefore, there is not a single perfect choice that can be used for all applications and ASD systems need to be carefully designed depending on the demands of a specific application.

Similarly to the recording duration, it is advisable to also fix a sampling rate that is sufficiently high for capturing normal and anomalous sounds when designing the ASD system for a specific task. This is often directly connected to the acoustic sensor, i.e. microphone to be used and may require expert knowledge. If using different sampling rates, one can decide for a fixed sampling rate and up- or downsample all recordings to this sampling rate. In case particular frequency bands completely contain all anomalous sounds, spectral filtering, e.g. with band-pass filters, can be applied to remove unimportant information and simplify the ASD task.

2.2.2 Spectral features

Waveforms are high-dimensional and thus many training samples are needed for training a model for ASD or any other non-trivial task [62]. This is the reason why the dimension of waveforms is often reduced before providing them as input for neural networks. Usually, this is accomplished by computing features based on classical signal processing. The most commonly used features are time-frequency representations such as magnitude spectrograms. These features consist of a frequency and time dimension and thus techniques used in image processing, as for example two-dimensional convolutional neural networks, can be utilized for computing the embeddings. Due to Heisenberg’s uncertainty principle, time and frequency resolution cannot simultaneously be chosen to be arbitrarily high. Increasing the frequency resolution degrades the time resolution and vice versa. Therefore, both resolutions have to be balanced and parameter settings have to be adjusted such that the features capture all sounds of interest for a particular application sufficiently well. Apart from using standard time-frequency representations obtained by applying a short-time Fourier transform (STFT), there are many other hand-crafted features with adjustable resolutions such as Mel-frequency scaled spectrograms [211], Mel-frequency cepstral coefficients [36] or features based on perceptual linear prediction [79]. One can also use a combination of features with different resolutions or wavelets [34]. Moreover, multiple hand-crafted features capturing complementary information can be combined to utilize more information in total and improve the performance [236]. This is called ensembling and will be discussed in Section 2.9. For ASD, application-dependent a priori knowledge about the anomalies, needed to design highly sophisticated features, is only rarely available as this requires to analyze anomalous samples, which are usually not available for training. One of the major strengths of neural networks is the ability to learn meaningful data representations by themselves. To utilize this strength, input feature representations that still capture most information present in the original data should be chosen instead of highly processed features.

The vast majority of state-of-the-art ASD systems for machine condition monitoring utilize log-Mel spectrograms as input features with no adaptations to specific machine types [21, 25, 59, 60, 85, 95, 164, 184, 186, 187, 212]. There are only a few minor deviations from this: In [123, 179], a Gammatone filterbank and in [39, 134, 174] regular spectrograms are used instead of or in combination with log-Mel spectrograms. For some systems [39, 112, 135], parameters for extracting log-Mel spectrograms such as window size, hop length and number of Mel bins are manually optimized for different machine types and data domains. In [124], different non-uniform filterbanks are determined for each machine type using Fisher’s F-ratio. Again, optimizing these parameters requires access to anomalous data, which is not available in a semi-supervised ASD setting and thus is infeasible. In [140], it has been shown that applying high-pass filters is beneficial to detect anomalies

in machine condition monitoring showing that high-frequency information is more important than low-frequency information.

2.2.3 Normalization

To ensure that all dimensions of the input feature representations are scaled similarly, a normalization technique can be applied. Some ASD systems perform a standardization of the input features in cepstral domain by subtracting the mean and dividing by the standard deviation of all normal training samples [85, 95, 179]. In [184], the waveforms are rescaled to unit variance and in [135] a Teager-Kaiser energy operator is used to suppress the noise for some machine types with modest improvements in ASD performance. Another simple denoising approach is to apply a median filter along the frequency axis [21]. Other works do not explicitly state a normalization procedure [21, 59, 60, 112, 174, 187] showing that a standardization of the input representations is an optional step. However, nearly all neural network based ASD approaches perform a standardization to all intermediate representations. This is called *batch normalization* [86] and results in a more stable and faster convergence of the training algorithm. To simplify notation for the further course of this thesis, \mathbf{X} will denote the space of appropriately pre-processed input feature representations instead of the raw waveforms unless stated otherwise.

2.3 ONE-CLASS EMBEDDINGS

The first type of embedding presented in this chapter is based on directly learning the distribution of normal data with a neural network. Using this distribution, one can compute a likelihood or distance of individual samples to distinguish between normal and anomalous samples. In general, this approach is called *inlier modeling* [96]. Since only a single class consisting of normal data is used, this is also known as *one-class classification* [156].

Among the models used for inlier modeling are deep generative models [17] such as autoregressive models [206], e.g. WaveNet [171] or a masked autoencoder for distribution estimation (MADE) [58], variational autoencoders (VAEs) [102], generative adversarial networks (GANs) [121], e.g. AnoGAN [201] or GANomaly [4], and normalizing flows [105, 172]. VAEs [102] can be seen as a generalization of autoregressive models [27] and thus a VAE with a sufficient number of hidden layers should, in theory, be able to achieve the same ASD performance as autoregressive models. Furthermore, it was shown that normalizing flows can be modified to outperform autoencoders such as VAEs [42, 43]. However, it is also known that normalizing flows have problems when detecting out-of-distribution samples, i.e. anomalies, [104, 161] and thus, by transitivity, VAEs and autoregressive models are also not an ideal choice. Moreover, deep autoregressive models [59, 75], normalizing flows [42, 43, 135, 184] or GANs [39, 90] do not aim at projecting data into an embedding space and thus are out of scope for this thesis. Hence, these ap-

proaches will not be discussed in detail and the focus will be on (non-variational) autoencoders [82] in general and deep support vector data description (SVDD) for one-class classification [196, 197], which has been shown to outperform at least some generative approaches for ASD, instead.

2.3.1 Autoencoders

In the context of this thesis, an autoencoder [82] is a model consisting of two sub-models: An embedding model, called encoder, mapping the input to an embedding space and an additional decoder model mapping an embedding back to the input space. The formal definition of an autoencoder is as follows.

Definition 2.1 (Autoencoder). Let \mathcal{W} denote the parameter space of a neural network $\varphi : \mathcal{X} \times \mathcal{W} \rightarrow \mathcal{X}$. Then, φ is called an *autoencoder* if it can be written as

$$\varphi = \varphi_d \circ \varphi_e \tag{1}$$

such that $\varphi_e : \mathcal{X} \times \mathcal{W}_e \rightarrow \mathbb{R}^D$, i.e. $\varphi_e \in \Phi$, and $\varphi_d : \mathbb{R}^D \times \mathcal{W}_d \rightarrow \mathcal{X}$ for parameter spaces \mathcal{W}_e and \mathcal{W}_d . φ_e and φ_d are called *encoder* and *decoder*, respectively. Embeddings $\varphi_e(x, w) \in \mathbb{R}^D$ obtained with an encoder for some $x \in \mathcal{X}$ and $w \in \mathcal{W}_e$ are also called *code*.

Autoencoders are trained by minimizing the reconstruction loss aiming at teaching the model to reconstruct input samples as accurately as possible. The reconstruction loss is given by the mean squared error (MSE) between the reconstructed input and the input signal and is defined as follows:

Definition 2.2 (Reconstruction loss). Let $\mathcal{Y} \subset \mathcal{X}$ be finite and let φ denote an autoencoder. Then, the *reconstruction loss* is defined as the mean squared error between input and output of the autoencoder

$$\begin{aligned} \mathcal{L}_{\text{rec}} &: \mathcal{P}(\mathcal{X}) \times \{\varphi | \varphi : \mathcal{X} \times \mathcal{W} \rightarrow \mathcal{X}\} \times \mathcal{W} \rightarrow \mathbb{R}_+, \\ \mathcal{L}_{\text{rec}}(\mathcal{Y}, \varphi, w) &:= \frac{1}{|\mathcal{Y}|} \sum_{x \in \mathcal{Y}} \|\varphi(x, w) - x\|_2^2. \end{aligned} \tag{2}$$

In order for an autoencoder to yield useful results, it needs to be ensured that the autoencoder learns a non-trivial mapping, i.e. not the identity mapping [64]. Otherwise, it holds that $\mathcal{L}_{\text{rec}} \equiv 0$. Hence, the embedding space does not need to have any structure and consequently embeddings of normal and anomalous samples do not need to be different. The most commonly used approaches for ensuring to learn a non-trivial mapping is to increase the difficulty of the task. A possible way to achieve this is to use a code (embedding) dimension that is much smaller than the dimension of the input space (undercomplete autoencoders) or to remove information from input samples, e.g. by randomly masking or modifying entries of the input vectors during training (denoising autoencoders) [5].

Conveniently, the reconstruction loss can also be used as an anomaly score by assuming that anomalous data substantially deviates from normal data. Then, an autoencoder trained to encode and reconstruct normal data cannot reconstruct anomalous data as accurately as normal data and thus yields a higher reconstruction error for anomalous data. However, in noisy environments an autoencoder cannot distinguish between noise and the target sounds and thus both signal components contribute to the reconstruction error with equal weight. This degrades performance, e.g. by causing false alarms, when trying to detect anomalous sounds in noisy environments such as factories.

There are several extensions and variants of basic autoencoders used for ASD. Most autoencoders encode a few consecutive frames of log-Mel spectrograms to reduce the dimension of the input to a manageable size. Usually, sub-classes of the normal data are defined and several autoencoders, one for each sub-class, are trained leading to specialized and thus more accurately modeled distributions of normal data [108]. Another idea is to use a single autoencoder for all classes that is trained to have a low reconstruction error when being conditioned on the correct sub-class and a high reconstruction error when being conditioned on another sub-class. In [95], this is accomplished by using the IDs of specific machines as class labels in the context of machine condition monitoring. In [112], the class labels of the input samples are also encoded and decoded and in [9, 179] a reconstruction loss as well as a classification loss are jointly minimized by using a (weighted) sum of both losses. [186] uses a single hierarchical VAE making use of hierarchical class labels such as machine types and model IDs, which are concatenated with the input feature representations as well as with latent representations. Furthermore, there are different asymmetric approaches regarding input and output of the autoencoder, which all improve ASD performance. In [212], an interpolation deep neural network has been proposed that reconstructs only the center frame of several consecutive frames taken from a log-Mel spectrogram with an autoencoder to remove the effect of the edge frames on the anomaly score, which are difficult to reconstruct. [59] relies on MADE [58] for density estimation, which applies binary masks to the input such that the autoencoder is an autoregressive (or a time-reversed autoregressive) model. Here, the autoencoder does not learn to reconstruct input data but learns conditional distributions, parameterized with Gaussians or a mixture of Gaussians, to predict the likelihood of the input for a specific time step while only depending on the input of previous (or subsequent) time steps. Then, the model is trained by minimizing the negative log-likelihood of the data given by the product of the learned conditional distributions. [235] generalizes the previous approach using attentive neural processes [99], which divide the input spectrogram into a context and a complementary target set. This is achieved by masking random time frames or frequency bins and also encoding the positions of the context set for computing the likelihood of the estimated target set conditioned on the context set.

2.3.2 Compactness loss

When using autoencoders to obtain audio embeddings for ASD, decoding the code is actually not needed. It is sufficient to only train an encoder network by computing the MSE between the code of input samples and a center $\mathbf{c} \in \mathbb{R}^D$ of a hypersphere in the embedding space. Then, a model can be trained by minimizing the volume of the hypersphere containing all normal samples. In [197], it has been shown that such a one-class approach outperforms commonly used autoencoder architectures or GANs such as AnoGAN [201] when detecting anomalies. The compactness loss incorporating this idea is defined as follows:

Definition 2.3 (Compactness loss). Let $Y \subset X$ be finite. Then, the *compactness loss* is defined as

$$\begin{aligned} \mathcal{L}_{\text{comp}} &: \mathcal{P}(X) \times \mathbb{R}^D \times \Phi \times \mathcal{W} \rightarrow \mathbb{R}_+, \\ \mathcal{L}_{\text{comp}}(Y, \mathbf{c}, \phi, \mathbf{w}) &:= \frac{1}{|Y|} \sum_{x \in Y} \|\phi(x, \mathbf{w}) - \mathbf{c}\|_2^2. \end{aligned} \quad (3)$$

The vector $\mathbf{c} \in \mathbb{R}^D$ is called *center* and belongs to the weight parameters of the neural network.

Remark. The original definition of the compactness loss also includes an additional weight decay term for regularization [197]. Since using such a regularization term is not strictly necessary and can be used to complement any loss function, this term has been omitted here for the sake of simplicity. All presented theoretical results still hold.

After training, the (squared) Euclidean distance between the embedding of a given sample and the center can be utilized as an anomaly score: A greater distance indicates a higher likelihood for the sample to be anomalous. Similarly as for autoencoders, it needs to be ensured that deep one-class classification models do not learn a trivial solution for minimizing the compactness loss, which in this case corresponds to a parameter setting $\mathbf{w}_{\mathbf{c}} \in \mathcal{W}$ for a center $\mathbf{c} \in \mathbb{R}^D$ such that $\phi(x, \mathbf{w}_{\mathbf{c}}) = \mathbf{c}$ for all $x \in X$. It is of utmost importance to prevent that the model to be trained is able to learn such a trivial solution. Otherwise, it is impossible to differentiate between normal and anomalous embeddings because all samples are mapped to the same point. Hence, preventing to learn trivial solutions is one of the major difficulties to overcome when training a one-class ASD model with normal data only.

As shown in [197], three conditions can be identified that need to be fulfilled to learn a non-trivial mapping. For simplicity, it will be assumed that all networks $\phi \in \Phi_{\text{simple}}$ have the following structure: Let $L(\phi) \in \mathbb{N}$ denote the number of layers of network $\phi \in \Phi_{\text{simple}}$. Then, $H_l(\phi)$ for $l \in \{0, \dots, L(\phi)\}$ are hidden representation spaces that can either be vectors, matrices or tensors, each with a well-defined

ordering of the entries allowing to index individual entries of the hidden representations. Now, set $H_0(\phi) := X$ and for $l \in \{1, \dots, L(\phi)\}$ iteratively define

$$H_l(\phi) := \{\sigma_l(\mathbf{w}(l) \cdot \mathbf{h}_{l-1}(x) + \mathbf{b}_l) : \mathbf{h}_{l-1}(x) \in H_{l-1}(\phi), x \in H_0(\phi) = X\} \quad (4)$$

such that $H_{L(\phi)}(\phi) \subset \mathbb{R}^D$ where $\mathbf{h}_l(x) \in H_l(\phi)$ denotes the hidden representation at layer l for input sample $x \in X$, “ \cdot ” denotes a linear operator, which is either a matrix multiplication or a convolution, σ_l denotes the activation function of layer l , $\mathbf{w}(l)$ denotes the corresponding weight parameters (excluding the bias term) and \mathbf{b}_l denotes the bias term. Using a network with such a structure, the three conditions that need to be met to avoid learning trivial solutions of the compactness loss are captured by the following propositions:

Proposition 2.4. *Using the same notation as in 2.3, let $\mathbf{w}_0 \in W$ denote the parameter setting of $\phi \in \Phi_{\text{simple}}$ with all weights being equal to zero. Then, $\mathbf{c}_0 := \phi(x, \mathbf{w}_0) \in \mathbb{R}^D$ is a constant function in $x \in X$ meaning that $\mathcal{L}_{\text{comp}}(Y, \mathbf{c}_0, \phi, \mathbf{w}_0) \equiv 0$ for all $Y \in \mathcal{P}(X)$.*

Proof. Applying a linear operator, which for the considered neural networks can either be a matrix multiplication or a convolution, with all parameters of the operator being zero yields zero in the representation space regardless of the argument. Therefore, the output of each layer is constant, i.e. $\phi(x, \mathbf{w}_0) = \mathbf{c}_0$ for all $x \in X$. Hence, $\mathcal{L}_{\text{comp}}(Y, \mathbf{c}_0, \phi, \mathbf{w}_0) \equiv 0$. \square

This proposition shows that a center $\mathbf{c} \in \mathbb{R}^D$, which is adapted during training of the model, can be learned to be equal to \mathbf{c}_0 yielding a trivial solution for minimizing the compactness loss with $\mathbf{w} = \mathbf{w}_0$. Thus, when training a network by minimizing the compactness loss it needs to be ensured that $\mathbf{c} \neq \mathbf{c}_0$ by using a non-trainable center. The following proposition shows that using bias terms also allow the network to learn a trivial solution.

Proposition 2.5. *Using the same notation as in 2.3, if any layer of $\phi \in \Phi_{\text{simple}}$ has a bias term, then there is a $\mathbf{w}_c \in W$ such that $\phi(x, \mathbf{w}_c) = \mathbf{c}$ for all $x \in X$.*

Proof. Let l_{bias} be the index of the last layer of ϕ with a bias term $\mathbf{b}_{l_{\text{bias}}}$. Then, for $\mathbf{w}_c \in W$ with $\mathbf{w}_c(l_{\text{bias}}) = 0$, it holds that $\mathbf{h}_{l_{\text{bias}}}(x) = \sigma_{l_{\text{bias}}}(\mathbf{b}_{l_{\text{bias}}})$ for all $x \in X$, i.e. $\mathbf{h}_{l_{\text{bias}}}(x)$ is a constant function. Therefore, all subsequent parameters $\mathbf{w}_c(l)$ for $l > l_{\text{bias}}$ can be chosen such that $\phi(x, \mathbf{w}_c) = \mathbf{c}$ for all $x \in X$ if $\mathbf{c} \in \text{Im}(\phi)$. \square

The last proposition investigates the effect of using bounded activation functions.

Proposition 2.6. *Using the same notation as in 2.3, let $H_l(\phi), H_{l+1}(\phi)$ denote hidden representation spaces belonging to two connected layers of $\phi \in \Phi_{\text{simple}}$. Let $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ be a monotonic and bounded activation function with $\sup_{a \in \mathbb{R}} \sigma(a) = B \neq 0$ or $\inf_{a \in \mathbb{R}} \sigma(a) = B \neq 0$. If there is an $i \in \{1, \dots, \dim(H_l(\phi))\}$ such that $\mathbf{h}_l^{(i)} > 0$ or $\mathbf{h}_l^{(i)} < 0$ for all $\mathbf{h}_l \in H_l(\phi)$, i.e. one component of the hidden representation space has always the same sign. Then, there is a parameter setting $\mathbf{w}_c \in W$ such that $\phi(x, \mathbf{w}_c) = \mathbf{c}$ for all $x \in X$.*

Proof. Without loss of generality assume $\sup_{\mathbf{a} \in \mathbb{R}} \sigma(\mathbf{a}) = \mathbf{B} \neq \mathbf{0}$ and $\mathbf{h}_l^{(i)} > \mathbf{0}$ for all $\mathbf{h}_l \in H_l(\phi)$ and some $i \in \{1, \dots, \dim(H_l(\phi))\}$. Then, by choosing $\mathbf{w}_c \in \mathcal{W}$ such that $\mathbf{w}_c(\mathbf{l})^{(j,k)} = \mathbf{0}$ for $j \in \{1, \dots, \dim(H_{l+1}(\phi))\}$ and $k \neq i$, we obtain

$$|\mathbf{h}_{l+1}^{(j)} - \mathbf{B}| = \left| \sigma \left(\sum_{k=1}^{\dim(H_l(\phi))} \mathbf{w}_c(\mathbf{l})^{(j,k)} \mathbf{h}_l^{(k)} \right) - \mathbf{B} \right| = \left| \sigma \left(\mathbf{w}_c(\mathbf{l})^{(j,i)} \underbrace{\mathbf{h}_l^{(i)}}_{>0} \right) - \mathbf{B} \right| < \epsilon$$

for all $\epsilon > \mathbf{0}$ and $\mathbf{w}_c(\mathbf{l})^{(j,i)}$ large enough. Hence, $\mathbf{h}_{l+1}^{(j)} = \mathbf{B}$ is constant. Because this emulates a bias term for layer $H_{l+2}(\phi)$, using Proposition 2.5 finishes the proof. \square

Remark. Note that using the commonly used rectified linear unit (ReLU) as an activation function is still possible because Proposition 2.6 only holds for activation functions with an upper or lower bound not equal to zero.

To sum up the previous propositions, a network trained by minimizing the compactness loss should use a non-trainable center that is not equal to $\mathbf{c}_0 = \phi(\mathbf{x}, \mathbf{w}_0)$ as defined in Proposition 2.4, not use any bias terms and not use bounded activation functions to avoid learning a trivial mapping. These propositions also hold in case the network contains pooling or normalization layers because these layers do not utilize any weights and zero is a fixed point if the input is constant. Hence, most commonly used convolutional neural network architectures used as embedding models are captured by these statements.

Another way to prevent the model from learning a constant function is to impose an additional classification task. In [177], an additional *descriptiveness loss* is used whose goal it is to reduce inter-class similarity between classes of an arbitrary, external multi-class dataset, which is only used to regularize the one-class classification task. This is done by minimizing the categorical cross-entropy loss for classification on this additional dataset as a secondary task. For each of the two tasks, another version of the same network with identical structure and tied weights, i.e. having the same weight settings, is used. During training, both losses are jointly minimized using a weighted sum ensuring that the so-called reference network associated with the compactness loss does not learn a constant function because this would prevent the secondary network to be able to classify correctly. In [113, 114], a classification task is defined by using available meta information as classes. During training, a second term consisting of the multiplicative inverse of the compactness loss for the center of all normal samples not belonging to the target class is used to avoid learning a trivial solution.

Note that autoencoders can also be viewed as a way to regularize one-class models. In this context, the encoder is the one-class model mapping the input to an embedding space. Unless the input space \mathbf{X} consists of a single point, learning a constant function mapping everything to the center is not an optimal solution for the reconstruction loss because all necessary information for being able to completely reconstruct an arbitrary input sample needs to be encoded, which requires obtaining different embeddings for different input samples.

2.4 AUXILIARY TASK EMBEDDINGS

Instead of directly modeling the distribution of the data to learn a mapping into an embedding space, one can also train a neural network to solve other tasks, so-called *auxiliary tasks*, not directly related to ASD. The underlying assumption is that, to solve an auxiliary task, specific information of the data needs to be captured in the embedding space and that this information is also sufficient to differentiate between normal and anomalous samples. Usually, classification tasks are used for this purpose. By essentially treating the other classes as pseudo anomalies, decision boundaries are learned for the normal data of each class. The difficulty of this approach is that one has to define suitable classes. For machine condition monitoring, possible auxiliary tasks are classifying between machine types [60, 85, 134, 276] or, additionally, between different machine states and noise settings [39, 168, 218], recognizing augmented and non-augmented versions of normal data (self-supervised learning) [60] or predicting the activity of machines [168]. Using an auxiliary task to learn embeddings is also called outlier exposure (OE) [78] because normal samples belonging to other classes than a target class can be considered proxy outliers [184].

To compute the output for many classification tasks solved by neural networks, the softmax function is used while minimizing the categorical crossentropy as a loss function for training. But when training a neural network for the purpose of extracting embeddings, the softmax function only reduces inter-class similarity without explicitly increasing intra-class similarity [223]. To address this issue, losses based on the Euclidean distance as for example triplet loss [233] and center loss [234] have been proposed. The triplet loss uses an anchor input whose distance to a positive sample belonging to the same class is minimized and whose distance to a negative sample belonging to another class maximized. Center loss avoids constructing these triplets as input for training by minimizing the distance to learned center vectors for each class. Recently, loss functions ensuring a margin between different classes such as additive margin softmax layer [223], SphereFace [133], CosFace [224], ArcFace [38] or AdaCos [274] were shown to have better generalization capabilities than losses based on the Euclidean distance because these losses enforce a low intra-class variability. For machine condition monitoring, [85] uses center loss, [134, 135] the additive margin softmax loss and [10, 39, 60, 112, 154, 276] use ArcFace. All of these losses lead to embeddings with a significantly better ASD performance than loss functions without a margin. Therefore, only angular margin losses will be discussed in detail.

2.4.1 Angular margin losses

The underlying idea of angular margin losses is to learn an embedding space on the unit sphere and compare two embeddings by calculating the angle between them while ensuring a margin between classes instead of only ensuring linear separability.

One of the most commonly used representative of angular margin losses is ArcFace [38], which will now be reviewed.

Definition 2.7 (ArcFace). Let $Y \subset X$ be finite and $\text{lab}(x)_j \in [0, 1]$ denote the j -th component of the categorical class label function $\text{lab} \in \Lambda(\mathbb{N}_{\text{classes}})$ where Λ denotes the space of all functions $\text{lab} : X \rightarrow [0, 1]^{\mathbb{N}_{\text{classes}}}$ with $\sum_{j=1}^{\mathbb{N}_{\text{classes}}} \text{lab}(x)_j = 1$ for all $x \in X$. Let $\text{softmax} : \mathbb{R}^{\mathbb{N}_{\text{classes}}} \rightarrow [0, 1]^{\mathbb{N}_{\text{classes}}}$ denote the softmax function, i.e.

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{\mathbb{N}_{\text{classes}}} \exp(x_j)}. \quad (5)$$

Then, the *ArcFace* loss is defined as

$$\begin{aligned} \mathcal{L}_{\text{ang}} &: \mathcal{P}(X) \times \mathbb{R}^{\mathbb{N}_{\text{classes}} \times D} \times \Phi \times W \times \Lambda(\mathbb{N}_{\text{classes}}) \times \mathbb{R}_+ \times [0, \frac{\pi}{2}] \rightarrow \mathbb{R}_+ \\ \mathcal{L}_{\text{ang}}(Y, C, \phi, w, \text{lab}, s, m) \\ &:= -\frac{1}{|Y|} \sum_{x \in Y} \sum_{j=1}^{\mathbb{N}_{\text{classes}}} \text{lab}(x)_j \log(\text{softmax}(s \cdot \text{sim}_{\text{mar}}(\phi(x, w), c_j, m))) \end{aligned} \quad (6)$$

where $C = (c_1, \dots, c_{\mathbb{N}_{\text{classes}}})$ and, in this case,

$$\begin{aligned} &\text{softmax}(s \cdot \text{sim}_{\text{mar}}(\phi(x, w), c_i, m)) \\ &:= \frac{\exp(s \cdot \text{sim}_{\text{mar}}(\phi(x, w), c_i, m))}{\sum_{j=1}^{\mathbb{N}_{\text{classes}}} \exp(s \cdot \text{sim}_{\text{mar}}(\phi(x, w), c_j, m \cdot \text{lab}(x)_j))} \end{aligned} \quad (7)$$

with

$$\text{sim}_{\text{mar}}(x, y, m) := \cos(\arccos(\text{sim}(x, y)) + m) \quad (8)$$

and

$$\text{sim}(x, y) := \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2} \in [-1, 1]. \quad (9)$$

The vectors $c_j \in \mathbb{R}^D$ are trainable parameters called *class centers*. The *margin* $m \in [0, \frac{\pi}{2}]$ and *scale parameter* $s \in \mathbb{R}_+$ are both hyperparameters that need to be chosen in advance.

The following intuitive explanation of why the parameter m ensures a margin between classes is similar to the one given in [223]. Define the angle α between an embedding $\phi(x, w) \in \mathbb{R}^D$ and a class center $c \in \mathbb{R}^D$ as

$$\alpha(\phi(x, w), c) := \arccos(\text{sim}(\phi(x, w), c)) \in [0, \pi].$$

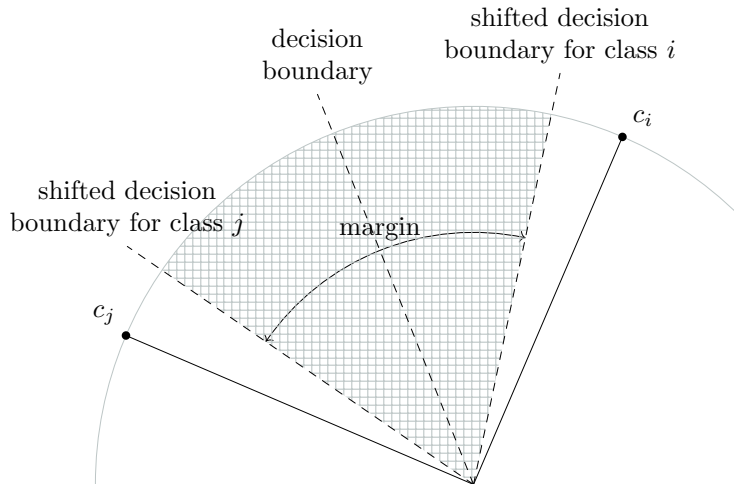


Figure 5: Ensuring an angular margin between classes i and j by shifting the decision boundaries. This illustration is inspired by Figure 1 from [223].

Since the softmax function is componentwise strictly monotonically increasing, the following inequality needs to be fulfilled to predict class $i \in \{1, \dots, N_{\text{classes}}\}$ for a given sample $x \in X$

$$\alpha(\phi(x, w), c_i) + m < \alpha(\phi(x, w), c_j) \text{ for all } j \neq i.$$

Hence, for $m = 0$ the decision boundary between two classes is the bisector of the angle between both corresponding class centers. Thus, only linear separability between the classes is ensured. For $m > 0$, the decision boundary is effectively shifted closer to c_i , which consequently reduces intra-class variability. Since *all* decision boundaries between two classes are shifted towards their corresponding class centers and thus the inequality does not depend on the choice of i , a margin depending on the chosen hyperparameter m is ensured. An illustration of this explanation can be found in Figure 5.

In [274], it has been shown that the choice of both hyperparameters, the scale parameter s and the margin m , have a significant impact on the posterior probabilities and thus also on the resulting performance. This is illustrated in Figure 6. On the one hand, a scale parameter that is too small limits the maximum posterior probability that can be achieved. This hinders the convergence of training because the model parameters will also be updated in case the angles between embeddings and class centers are already very small. Similarly, a margin that is too large impedes convergence of the training procedure because the posterior probabilities are almost equal to zero even when the angles are very small. On the other hand, a scale parameter that is too large or a margin that is too small lead to the posterior probabilities being equal to one, even for large angles, and thus the model is not sensitive to changes of the angle below a certain point. Therefore, the effect on the sensitivity of the output with respect to the angle is similar when strongly varying the magnitude for both of these parameters and a single appropriately

chosen parameter is sufficient for controlling the posterior probabilities. Moreover, it has been shown experimentally that an adaptive scale parameter outperforms using two tuned, but fixed, parameters [274].

Following the main line of argumentation as presented in [274], it will now be briefly explained how the definition of this so-called *dynamically adaptive scale parameter* is motivated. For fixed $\mathbf{x} \in \mathbf{X}$, the scale parameter should be chosen such that it maximizes the sensitivity of the softmax probability with respect to the corresponding angle. Formally, this means to find the scale parameter $s_0 \in \mathbb{R}_+$ that fulfills

$$\frac{\partial^2}{\partial \alpha_0^2} \frac{\exp(s_0 \cdot \cos(\alpha_0))}{\exp(s_0 \cdot \cos(\alpha_0)) + \sum_{\substack{j=1 \\ \text{lab}(\mathbf{x})_j \neq 1}}^{\text{N}_{\text{classes}}} \exp(s_0 \cdot \sin(\phi(\mathbf{x}, \mathbf{w}), \mathbf{c}_j))} = 0,$$

which is a transcendental equation approximately equivalent to

$$s_0 = \frac{\log \left(\sum_{\substack{j=1 \\ \text{lab}(\mathbf{x})_j \neq 1}}^{\text{N}_{\text{classes}}} \exp(s_0 \cdot \sin(\phi(\mathbf{x}, \mathbf{w}), \mathbf{c}_j)) \right)}{\cos(\alpha_0)}.$$

By conducting experimental evaluations, it can be seen that $\alpha(\phi(\mathbf{x}, \mathbf{w}), \mathbf{c}_j) \approx \frac{\pi}{2}$, i.e. $\phi(\mathbf{x}, \mathbf{w})$ and \mathbf{c}_j are approximately orthogonal, and thus $\sin(\phi(\mathbf{x}, \mathbf{w}), \mathbf{c}_j) \approx \cos\left(\frac{\pi}{2}\right) = 0$ for all j with $\text{lab}(\mathbf{x})_j = 0$. Therefore, choosing a fixed scale parameter $\tilde{s}^{(0)} \in \mathbb{R}_+$ is the same as choosing a fixed value $\alpha_0^{(0)} \in [0, \frac{\pi}{2}]$ in the equation above. A natural choice for ensuring that the scale parameter is neither too high nor too low is setting $\alpha_0^{(0)}$ to the center of this interval, i.e. $\alpha_0^{(0)} = \frac{\pi}{4}$, resulting in

$$\tilde{s}^{(0)} = (\cos\left(\frac{\pi}{4}\right))^{-1} \cdot \log \left(\sum_{\substack{j=1 \\ \text{lab}(\mathbf{x})_j \neq 1}}^{\text{N}_{\text{classes}}} \exp(s_0 \cdot \cos\left(\frac{\pi}{2}\right)) \right) = \sqrt{2} \cdot \log(\text{N}_{\text{classes}} - 1).$$

During training, the angle $\alpha(\phi(\mathbf{x}, \mathbf{w}), \mathbf{c}_i)$ is decreasing, which also decreases the sensitivity of the softmax probability to further changes of this angle (see [Figure 6](#)). To avoid this and preserve the same sensitivity, the scale parameter should also be decreased. By monitoring the median of all angles between the training samples and their corresponding class centers at training step $t \in \mathbb{N}_0$, denoted by $\alpha_{\text{med}}^{(t)} \in [0, \frac{\pi}{2}]$, the training progress can be monitored, too. However, initially these angles may be relatively large. To increase sensitivity in these early iterations, the median angle is forced to be smaller than $\frac{\pi}{4}$ by setting $\alpha_0^{(t)} = \min\{\frac{\pi}{4}, \alpha_{\text{med}}^{(t)}\}$. Furthermore, the term

$$\sum_{\substack{j=1 \\ \text{lab}(\mathbf{x})_j \neq 1}}^{\text{N}_{\text{classes}}} \exp(\tilde{s}^{(t-1)} \cdot \sin(\phi(\mathbf{x}, \mathbf{w}), \mathbf{c}_j))$$

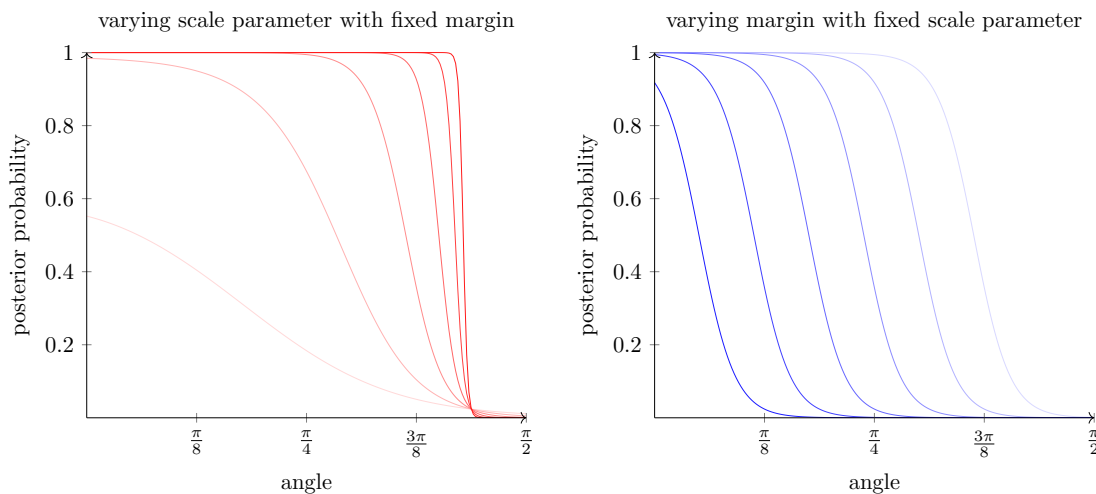


Figure 6: Effects of varying the margin and the scale parameter of an angular margin loss on the relationship between angle and posterior probability. Brighter colors indicate smaller parameters and darker colors indicate higher parameters. This illustration is based on Figure 2 from [274].

is computed for each sample of a mini-batch and the mean, denoted by $B_{\text{avg}}^{(t)}$, is taken to more accurately reflect the current training progress. In conclusion, the dynamically adaptive scale parameter for $t > 0$ is set to

$$\tilde{s}^{(t)} = \frac{\log B_{\text{avg}}^{(t)}}{\cos(\alpha_0^{(t)})} = \frac{\log B_{\text{avg}}^{(t)}}{\cos(\min(\frac{\pi}{4}, \alpha_{\text{med}}^{(t)}))}.$$

Using this dynamically adaptive scale parameter, the AdaCos loss [274] is defined as follows.

Definition 2.8 (AdaCos). Using the same notation as in Definition 2.7, let $Y^{(t)} \subset Y$ denote all samples belonging to a batch of size $N_{\text{batch}} \in \mathbb{N}$, i.e. $|Y^{(t)}| = N_{\text{batch}}$. Let $\alpha(\phi(x, w), c) := \arccos(\text{sim}(\phi(x, w), c)) \in [0, \pi]$ denote the angle between $\phi(x, w) \in \mathbb{R}^D$ and $c \in \mathbb{R}^D$. Further, let the *dynamically adaptive scale parameter* $\tilde{s}^{(t)} \in \mathbb{R}_+$ at training step $t \in \mathbb{N}_0$ be defined as

$$\tilde{s}^{(t)} := \begin{cases} \sqrt{2} \cdot \log(N_{\text{classes}} - 1) & \text{if } t = 0 \\ \frac{\log B_{\text{avg}}^{(t)}}{\cos(\min(\frac{\pi}{4}, \alpha_{\text{med}}^{(t)}))} & \text{else} \end{cases} \quad (10)$$

where $\alpha_{\text{med}}^{(t)} \in [0, \frac{\pi}{2}]$ denotes the median of all angles $\alpha(\phi(x, w), c_{\text{class}(x)})$ with $x \in Y^{(t)}$ and $\text{class}(x) \in \{1, \dots, N_{\text{classes}}\}$. For $w^{(t)} \in W$,

$$B_{\text{avg}}^{(t)} := \frac{1}{N_{\text{batch}}} \sum_{x \in Y^{(t)}} \sum_{\substack{j=1 \\ \text{lab}(x)_j \neq 1}}^{N_{\text{classes}}} \exp(\tilde{s}^{(t-1)} \cdot \text{sim}(\phi(x, w^{(t)}), c_j)) \quad (11)$$

is the sample-wise average over all summed logits belonging to the non-corresponding classes. Then, the *AdaCos* loss is defined as

$$\begin{aligned} \mathcal{L}_{\text{ada}} &: \mathcal{P}(\mathcal{X}) \times \mathbb{R}^{\text{N}_{\text{classes}} \times \text{D}} \times \Phi \times \mathcal{W} \times \Lambda(\text{N}_{\text{classes}}) \rightarrow \mathbb{R}_+ \\ \mathcal{L}_{\text{ada}}(Y, C, \phi, w, \text{lab}) &:= \mathcal{L}_{\text{ang}}(Y, C, \phi, w, \text{lab}, \tilde{s}, 0). \end{aligned} \quad (12)$$

2.4.2 Handling imbalanced data

When using an auxiliary classification task, it may be the case that the numbers of training samples for each of the considered classes are highly imbalanced. This leads to a learned bias towards specific classes when training a model, which is usually not favourable for computing embeddings and thus should be avoided. There are several techniques to counteract this effect [92, 214] as for example implemented in the *imbalanced-learn* package [119]. In general, one can distinguish between two major paradigms: Data-level methods and algorithm-level methods. Data-level approaches assign different likelihoods to individual samples for being used as training samples and can be divided into oversampling and undersampling methods. For oversampling, random training samples belonging to classes with fewer training samples are used multiple times until there are as many samples for each class as for the class with the highest number of training samples. For undersampling, only as many random training samples as belonging to the class with the fewest samples are used for each class. There are also more sophisticated sampling methods such as the synthetic minority over-sampling technique ([SMOTE](#)) [24]. Here, existing training samples of the minority classes are not simply sampled uniformly at random but linear interpolations between training samples and one of their nearest neighbors are used as synthetic training samples to balance the number of samples per class. Algorithm-level approaches use weights or so-called importance factors for different classes with respect to the number of samples available and apply them during training. One example is weighting the loss associated with each sample such that samples belonging to classes with fewer training samples have a greater contribution to the gradient for updating the parameters of a neural network. Another example is the Focal loss [129], which assigns more weight to samples that are not classified correctly during training.

2.5 PRE-TRAINED EMBEDDINGS

Instead of using an auxiliary task for training the embedding model on an application-dependent dataset, one can also utilize such an auxiliary task on an external dataset [190]. Usually, large datasets are used for this purpose and the resulting embeddings are called *pre-trained* embeddings. The assumption is that the learned embeddings have encountered many diverse data samples and thus should also encode information useful for detecting anomalies on a small application-dependent dataset. Using pre-trained embeddings is especially promising when

the number of training samples for the actual application is very small, making it difficult to learn useful representations. Another advantage of using pre-trained embeddings is that these embeddings can be used for multiple applications or novel sound classes without the need of specifically training an individual model for each task from scratch.

There are several systems for ASD [68, 239] and OSC [162, 237] based on pre-trained audio embeddings and studies comparing these embeddings for ASD [158] or audio classification tasks [67] in settings with sufficient training data. Another approach is to use image embeddings for ASD [159] or to apply them for zero-shot audio classification [41]. In [166], it is shown that combining multiple hidden representations of pre-trained neural networks improves the performance. A few commonly used pre-trained embeddings will now be briefly reviewed:

- *VGGish* [80] is a modified version of the VGG network [207], named after the Visual Geometry Group from the University of Oxford, with a similar architecture. The network is pre-trained in a supervised manner on a preliminary version of YouTube-8M [2], which consists of 2.6 billion audio segments from YouTube videos belonging to a total of 3628 classes. The resulting embeddings have a feature dimension of 128 with an additional time dimension resulting from a sliding window of 960 ms with no overlap applied to the waveforms.
- *Kumar* embeddings [111] are extracted using a convolutional neural network (CNN) with a VGG-style architecture [207]. The network is pre-trained in a supervised manner on the so-called balanced subset of AudioSet [57] that consists of around 22,000 audio clips from YouTube videos belonging to 527 sound classes. The resulting embeddings have a feature dimension of 1024 and no time dimension because of a global temporal pooling operation inside the network.
- *OpenL3* [32] is a network trained to extract *look, listen, and learn (L3)* embeddings [6, 7]. There are multiple versions of this network: One is pre-trained on a music subset and the other one on an environmental subset of AudioSet [57] consisting of 296K and 195K YouTube videos, respectively. The network is trained in a self-supervised manner to decide whether a video frame and the log-spectrogram of an audio clip with a length of one second do or do not belong together using a convolutional audio and a convolutional video sub-network. During training, the embeddings of both sub-networks are concatenated and further processed with a binary classification module consisting of two fully-connected and a softmax layer. After training, only the audio sub-network is needed to extract embeddings from audio data. The resulting embeddings have a feature dimension of 512 with an additional time dimension resulting from a sliding window of one second with a hop size of 0.1 seconds applied to the waveforms.

- *Pre-trained audio neural network (PANN)* [110] is a combination of a one-dimensional sub-network applied to waveforms (Wavegram-CNN) and a two-dimensional sub-network applied to log-Mel spectrograms. Both output representations are concatenated and further processed with another two-dimensional sub-network. The entire network is pre-trained in a supervised manner on AudioSet [57] using a total of 1,934,187 audio clips from YouTube videos belonging to 527 sound classes. As the difference in performance between including and not including Wavegram-CNN is relatively small, only the sub-network with a VGG-like architecture pre-trained on log-Mel spectrograms (CNN14) is used for the experiments conducted in this thesis. The resulting embeddings have a feature dimension of 2048 with no time dimension because of a global temporal pooling operation inside the network.

2.6 COMPUTING AN ANOMALY SCORE

After projecting audio data into a suitable embedding space, the next step in the ASD processing chain is to calculate an anomaly score. As stated in Chapter 1, an anomaly score of a sample $x \in X$ is a value $\text{score}(x) \in \mathbb{R}$ indicating how likely this sample is to be anomalous with higher values corresponding to higher likelihoods. To achieve this, a given sample x is first projected into the embedding space \mathbb{R}^D by applying a neural network $\phi \in \Phi$ with parameters $w \in W$. Afterwards, a scoring function $\text{score}_{\text{emb}} : \mathbb{R}^D \rightarrow \mathbb{R}$ is applied to the embedding. Hence, the anomaly score is actually a composition of two functions, i.e. $\text{score} = \text{score}_{\text{emb}} \circ \phi$ with $\text{score}(x, w) = \text{score}_{\text{emb}}(\phi(x, w))$. This section is dedicated to presenting several possible choices for setting $\text{score}_{\text{emb}}$.

Depending on the type of model used for extracting the embeddings, there are often canonical choices for computing an anomaly score: For models trained with one-class losses, the negative sample-wise loss, e.g. the reconstruction error [33, 95, 123, 174, 186, 212, 235] or the negative log-likelihood [42, 43, 59, 235] of a single sample, can be utilized as an anomaly score. For models trained with an auxiliary classification task, the negative posterior probabilities [9, 11, 21, 60, 85, 134, 183, 232], or the cosine distance (CD) between an extracted embedding and the learned class centers can be used [10, 25, 26, 39, 134, 217, 218, 269, 276]. Other works utilize the Mahalanobis distance [39, 269] or the Euclidean distance [113, 114, 164].

It is also possible to train an additional model on the set of normal embeddings to estimate their distribution and being able to calculate an anomaly score by using this model. In [96], this is called a *sequential approach* because first an outlier-exposed model is used to extract embeddings and then inlier modeling is applied to obtain an anomaly score. Examples of such models are a Gaussian mixture model (GMM) [68, 76, 114, 115, 155, 158, 159, 187], local outlier factor (LOF) [10, 39, 114, 115, 154, 155], k-nearest neighbors (k-NN) [39, 115, 154, 155, 164,

219, 276] or probabilistic linear discriminant analysis (PLDA) [185] as done in [239]. Using such a sequential approach is especially important for pre-trained embeddings, since there is not always a canonical choice for computing an anomaly score as the training process of the embedding model is usually not related to the ASD task. In [159], the ASD performances obtained when using a GMM, a one-class support vector machine (OCSVM), an isolation forest, a Bayesian GMM, a kernel density estimator or an autoencoder for pre-trained embeddings have been compared and concluded that using a GMM or OCSVM yield the best results. Often, the dimension of pre-trained embeddings is reduced by applying techniques such as principal component analysis (PCA) [68, 158, 239] or linear discriminant analysis (LDA) [239] before inserting them into one of the previously mentioned models for calculating an anomaly score.

Note that many ASD systems utilize multiple ways for computing an anomaly score by choosing a different method for each class to optimize the ASD performance or by ensembling multiple metrics or models (see Section 2.9). Examples are to use the sum of the MSE and posterior classification probabilities [9] or using the frame-level reconstruction errors of a one-class model as input features for a GMM [76]. In [68], the dimension of the temporal mean of openL3 embeddings is reduced with PCA and the result is concatenated with reconstruction errors from autoencoders to use them as input features for a GMM.

2.7 DATA AUGMENTATION

To increase the generalization capabilities of a data-driven model, one should use as much proper training data as possible. However, available training data is usually limited and recording additional training data requires at least some effort or may not be feasible. A more efficient approach is to modify available training samples to simulate additional training data, which is called *data augmentation* [64]. In this section, the three data augmentation approaches most frequently used for ASD, namely mixup [272], SpecAugment [173] and different methods for simulating anomalous data, will be presented.

2.7.1 Mixup

Due to its simplicity and effectiveness, one of the most popular data augmentation techniques for audio data is *mixup* [272]. It is also used for several ASD systems [9–11, 21, 25, 60, 112, 164, 182]. The idea of mixup is to linearly interpolate between random training samples and their corresponding categorical class labels to generate new samples during training. Formally, mixup is defined as follows:

Definition 2.9 (Mixup). Let $x_1, x_2 \in X$ be random training samples. Then, for $\lambda \in [0, 1]$ with $\lambda \sim \text{Beta}(\beta_{\text{mix}}, \beta_{\text{mix}})$ the sample mix-

ing function $\text{mix}_x : X \times X \times [0, 1] \rightarrow X$ and the label mixing function $\text{mix}_{\text{lab}} : [0, 1]^{\text{N}_{\text{classes}}} \times [0, 1]^{\text{N}_{\text{classes}}} \times [0, 1] \rightarrow [0, 1]^{\text{N}_{\text{classes}}}$ are defined as

$$\begin{aligned} \text{mix}_x(x_1, x_2, \lambda) &= \lambda x_1 + (1 - \lambda)x_2, \\ \text{mix}_{\text{lab}}(x_1, x_2, \lambda) &= \lambda \text{lab}(x_1) + (1 - \lambda) \text{lab}(x_2). \end{aligned}$$

Usually, $\beta_{\text{mix}} = 1$ is used without any significant difference in the resulting performance and thus $\lambda \sim \mathcal{U}(0, 1)$. Also, note that

$$\begin{aligned} \sum_{j=1}^{\text{N}_{\text{classes}}} \text{mix}_{\text{lab}}(x_1, x_2, \lambda)_j &= \lambda \sum_{j=1}^{\text{N}_{\text{classes}}} \text{lab}(x_1)_j + (1 - \lambda) \sum_{j=1}^{\text{N}_{\text{classes}}} \text{lab}(x_2)_j \\ &= \lambda + (1 - \lambda) = 1. \end{aligned}$$

and therefore $\text{mix}_{\text{lab}}(x_1, x_2, \lambda) \in [0, 1]^{\text{N}_{\text{classes}}}$ is a valid categorical class label.

There are several variants of mixup. For machine condition monitoring, it has been proposed to only mix samples of individual machines or parameter settings belonging to the same machine type [25]. Mixup can also be applied to intermediate representations of a network instead of the input representations. This modification is called manifold mixup [220]. One can also combine two samples in multiple other ways than using linear interpolations as for example by concatenating different parts of two audio signals or spectral features. A collection of ways to mix samples can be found in [213].

2.7.2 SpecAugment

SpecAugment is the combined application of three different data augmentation techniques, namely *frequency masking*, *time masking* and *time warping*. Originally, SpecAugment [173] has been developed to modify time-frequency representations for automatic speech recognition (ASR) but is also used for ASD [9, 11, 112, 164, 219]. For frequency and time masking, random frequency bands or time frames of random size are masked with zeros. Usually, multiple time and frequency masks of relatively small size are used. For time warping, the time dimension is randomly split into two regions of which one is squeezed and the other is stretched with a random degree. An illustration of SpecAugment can be found in Figure 7. To adjust the effects of SpecAugment, manually tuned hyperparameters for controlling the number of time and frequency masks as well as their maximum size and a time warp parameter are used. In [9], masking randomly sized squares of time-frequency representations while training an autoencoder led to a better ASD performance than when applying SpecAugment, when only masking frequencies or when only masking time frames during training. Furthermore, some works used similar methods as SpecAugment such as shifting time frames in time [21] or spectral warping [60].

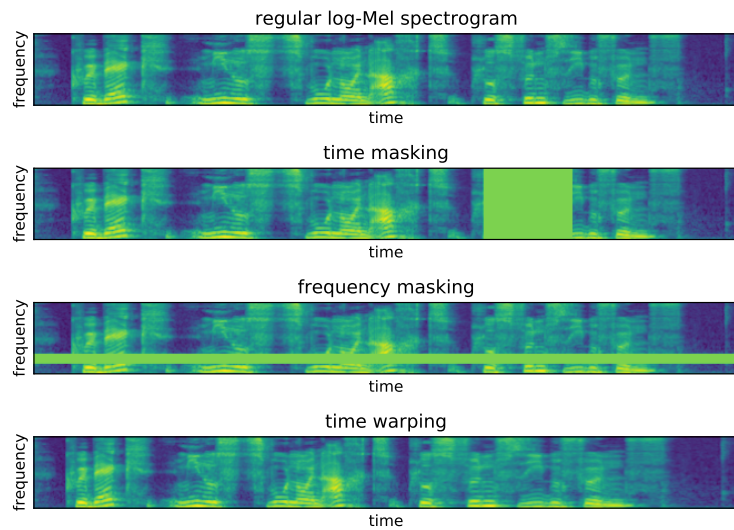


Figure 7: Illustration of the three different data augmentation techniques used when applying SpecAugment. This Figure strongly resembles Figure 1 contained in [173].

2.7.3 *Simulating anomalies*

A data augmentation approach with a completely different goal than creating additional normal samples is to *simulate anomalies* by modifying normal samples. However, to have a significant impact on the ASD performance simulated anomalies need to be realistic, i.e. very similar to real anomalies. Since acquiring specific knowledge about anomalies still requires access to anomalous training samples, this is in many cases out of scope for a specific semi-supervised ASD task. Still, there are some non-specific methods that can be applied to simulate anomalies for various tasks.

In the context of machine condition monitoring, normal data belonging to other machine IDs and machine types from the same dataset are used as proxy outliers and a binary classifier is trained for each machine ID [183, 184]. In [112], only normal data belonging to the same machine type but to other machine IDs is used to train a binary classifier. Similarly, [43] uses normal data belonging to other machine types as outliers for training a normalizing flow. Note that this is very similar to using an auxiliary classification task. The only difference is that another model is trained for each class instead of training a single model for all classes. Hence, all models trained to extract auxiliary task embeddings can also be viewed as utilizing data belonging to other classes as proxy-outliers without explicitly stating this and thus other normal sounds are implicitly used as pseudo-anomalies by many state-of-the-art ASD systems.

Another idea to simulate anomalies is to modify normal samples and treat these modified samples as if they belong to an additional anomalous class, which is a

Table 2: Advantages and disadvantages of different types of audio embeddings. The rating scale consists of ++ (very advantageous), + (advantageous), - (disadvantageous) and -- (very disadvantageous).

	one-class embeddings	auxiliary task embeddings	pre-trained embeddings
ASD performance	+	++	-
OSC and SED performance	-	++	-
data requirements	+	-	++
training effort	--	+	++
expandability	+	--	++
data augmentation possibilities	+	++	-
affected by imbalanced data	++	--	++
anomaly score computation	++	++	-
anomaly localization	+	-	--

self-supervised learning (SSL) approach. In [85], pitch shifting, time stretching and image transformations of the spectrograms are used to simulate anomalies. In [134] mixup with a fixed small mixing coefficient is used to create pseudo-anomalies. [26] proposes a method called *statistics exchange* and shows that this approach outperforms applying mixup when simulating anomalies. This method consists of swapping first- and second-order statistics of two normal samples for randomly chosen consecutive frequency bands or time frames. Utilizing SSL for training an embedding model will be discussed in more detail in Section 5.7. A more complex procedure to simulate anomalous samples is described in [107]. Here, the authors propose a rejection sampling algorithm that uses latent representations of an autoencoder and a GMM to generate anomalous sounds.

2.8 COMPARISON OF DIFFERENT EMBEDDING TYPES

To recapitulate the different types of embeddings presented in the previous sections, their advantages and disadvantages as summarized in Table 2 will now be discussed. When comparing the embeddings with respect to the resulting ASD performance, auxiliary task embeddings usually perform best [50, 72], followed by one-class embeddings. Pre-trained embeddings perform worst but still yield better results than random guessing. For OSC and SED tasks, the performances obtained with different embedding types are ranked similarly because the auxiliary task can be chosen as the classification task between the known classes of interest and thus the auxiliary task embeddings are well-suited for discriminating between the known classes. For one-class embeddings, the performance is slightly worse due to differently scaled anomaly scores obtained for each class making it difficult to determine the correct known class via maximum likelihood. When comparing the data requirements for training the different types of embeddings, pre-trained embeddings do not need any training data by definition but only a few normal samples, down to a single one, to be able to compare given test samples to them.

One-class embeddings need more normal samples for training the model. Training auxiliary task embeddings not only requires access to a sufficient number of training samples but also additional meta information to be used as class labels and thus have the strongest data requirements. One can also choose to use a self-supervised auxiliary task to artificially generate class labels, but the performance is usually worse when using manually created class labels. The effort of training a model is lowest for pre-trained embeddings because the model is already trained. For auxiliary task embeddings, a single model is trained for all normal classes whereas for one-class embeddings usually several models are trained, one for each normal sound event class, unless an extended approach consisting of training a single model for multiple classes is used. This also means that when expanding the ASD system with an additional normal sound event class, for single-class embeddings one can train an additional model for this class and, for the other classes, keep the remaining models as they were. For auxiliary task embeddings, the whole system needs to be re-trained. For pre-trained embeddings, no additional training is required. There are several possibilities to apply data augmentation techniques when training an ASD system based on auxiliary task embeddings as presented in Section 2.7. Some of these techniques can also be applied when training models based on one-class embeddings. However, for pre-trained embeddings the use of data-augmentation techniques is very limited unless one applies some form of transfer learning [265] because otherwise no training is taking place. Because of the same reason, pre-trained embeddings are not affected by imbalanced classes. Furthermore, imbalanced classes do not affect one-class embeddings because the class-specific models are trained independently of each other. Although there are techniques to counter the effects of imbalanced classes for auxiliary task embeddings, severely imbalanced classes are still a problem. To compute an anomaly score, for one-class and auxiliary task embeddings there are canonical as well as several other choices (see Section 2.6) whereas for pre-trained embeddings there are in general no canonical ways to calculate anomaly scores. Last but not least, at least some one-class models can be used for localizing anomalies in time, e.g. by utilizing a time-wise reconstruction error. When using auxiliary task embeddings, in general the only direct way to localize anomalies is to use a windowing approach, which often degrades the ASD performance. For many pre-trained embeddings, it is not even possible to choose a window size because the model has been trained with a fixed, relatively large, window size.

2.9 ENSEMBLING

Different models or the embeddings obtained with them correspond to different views on the input data. Since these different views may result in complementary information about the data, multiple models can be utilized in a single system. Such a system is called an *ensemble* [54]. Usually, an ensemble outperforms all individual sub-systems but inherently requires much more computational resources.

For [ASD](#) with audio embeddings, there are two main ensembling approaches: The first approach is to utilize multiple backends for a single embedding model [168, 198, 218] and the second approach is to train multiple sub-systems with different embedding models and combine their output [59, 60, 76, 85, 95, 112, 135, 219]. Often both approaches are applied at once, resulting in even larger ensembles [33, 39, 115, 232]. Another more complex ensembling approach is presented in [21]. Here, an autoencoder and a supervised classification model are combined with an additional contrastive loss between both resulting embeddings and the system is jointly trained by minimizing all three loss functions. Note that, in general, the higher the number of models used for an ensemble the smaller the obtained performance improvements while the computational overhead in terms of memory and time is constant and thus the return on investment is diminishing. Hence, while for academic purposes one may choose to use a huge ensemble to outperform other published [ASD](#) systems, for practical applications the size of an ensemble should be considered economically.

Usually, an ensemble is created by taking the mean of anomaly scores of several models or systems [59, 60, 85, 112, 115, 168, 198, 232]. To take differently scaled anomaly scores resulting from using different backends into account, the anomaly scores are often normalized before combining them by using the anomaly scores of normal training samples as reference values. Some systems use a weighted mean of (normalized) anomaly scores [33, 39, 76, 95, 135, 218, 219] with weights that are optimized to improve the [ASD](#) performance on a development dataset. Hence, these systems also use anomalous samples as training data, which, strictly speaking, is not allowed in a semi-supervised [ASD](#) setting. However, one could also argue that every [ASD](#) system is evaluated with some anomalous samples during development and therefore uses anomalous data for training. Moreover, one can raise the questions of whether it is possible at all to define the term *normal* without having access to application-dependent anomalous samples and, in conclusion, whether unsupervised or semi-supervised anomaly detection settings truly exist. Answering these questions is out of scope for this thesis as they appear to be of purely philosophical nature. Still, the dataset of an [ASD](#) task can be designed to prevent explicitly utilizing anomalous samples for tuning hyperparameters, e.g. by using data belonging to different machine types for developing and evaluating the system (see [Section 5.2](#)).

2.10 EVALUATION METRICS

Evaluating the performance of a trained system is important to be able to compare different design choices, optimize hyperparameters during the development of a system and allows to objectively compare different systems. For this purpose, specifically designed evaluation metrics are used. In this section several commonly used metrics will be reviewed for [ASD](#), [OSC](#) and [SED](#).

2.10.1 Anomaly detection

For ASD, several evaluation metrics can be used. In general, one can distinguish between *threshold-dependent* and *threshold-independent* evaluation metrics.

Threshold-dependent evaluation metrics are based on a fixed decision threshold $\theta \in \mathbb{R}$. When viewing anomaly detection as a binary classification problem with anomalies as positive samples and normal samples as negative ones, there are two errors that can occur for a test sample $x \in X_{\text{test}} \subset X = X_{\text{normal}} \cup X_{\text{anomalous}}$:

$$x \in X_{\text{normal}} \quad \text{but} \quad \text{score}(x) > \theta \quad (\text{false positive}) \quad (13)$$

or

$$x \in X_{\text{anomalous}} \quad \text{but} \quad \text{score}(x) \leq \theta \quad (\text{false negative}). \quad (14)$$

Let $A_{\text{pred}}(X_{\text{test}}, \text{score}, \theta) := \{x \in X_{\text{test}} : \text{score}(x) > \theta\} \subset X_{\text{test}}$ denote the set of predicted anomalies. Let X_{normal} and $X_{\text{anomalous}}$ be fixed. To count both errors for the entire test set X_{test} , we define the cardinality of the set of false positives (FP) and the set of false negatives (FN) as

$$\text{FP}(X_{\text{test}}, \text{score}, \theta) := |A_{\text{pred}}(X_{\text{test}}, \text{score}, \theta) \cap X_{\text{normal}}| \quad (15)$$

and

$$\text{FN}(X_{\text{test}}, \text{score}, \theta) := |(X_{\text{test}} \setminus A_{\text{pred}}(X_{\text{test}}, \text{score}, \theta)) \cap X_{\text{anomalous}}|. \quad (16)$$

Similarly, one can define the cardinality of the set of true positives (TP) and the set of true negatives (TN) as

$$\text{TP}(X_{\text{test}}, \text{score}, \theta) := |A_{\text{pred}}(X_{\text{test}}, \text{score}, \theta) \cap X_{\text{anomalous}}| \quad (17)$$

and

$$\text{TN}(X_{\text{test}}, \text{score}, \theta) := |(X_{\text{test}} \setminus A_{\text{pred}}(X_{\text{test}}, \text{score}, \theta)) \cap X_{\text{normal}}|. \quad (18)$$

Since the final evaluation metrics are based on these cardinalities, they are also called *intermediate statistics*. By normalizing these cardinalities, we obtain the corresponding error rates

$$\begin{aligned} \text{FPR}(X_{\text{test}}, \text{score}, \theta) &:= \frac{\text{FP}(X_{\text{test}}, \text{score}, \theta)}{|X_{\text{test}} \cap X_{\text{normal}}|} && \in [0, 1] \\ \text{FNR}(X_{\text{test}}, \text{score}, \theta) &:= \frac{\text{FN}(X_{\text{test}}, \text{score}, \theta)}{|X_{\text{test}} \cap X_{\text{anomalous}}|} && \in [0, 1] \\ \text{TPR}(X_{\text{test}}, \text{score}, \theta) &:= \frac{\text{TP}(X_{\text{test}}, \text{score}, \theta)}{|X_{\text{test}} \cap X_{\text{anomalous}}|} && \in [0, 1] \\ \text{TNR}(X_{\text{test}}, \text{score}, \theta) &:= \frac{\text{TN}(X_{\text{test}}, \text{score}, \theta)}{|X_{\text{test}} \cap X_{\text{normal}}|} && \in [0, 1]. \end{aligned} \quad (19)$$

Note that one can easily see that $\text{FPR}(\mathcal{X}_{\text{test}}, \text{score}, \theta) = 1 - \text{TNR}(\mathcal{X}_{\text{test}}, \text{score}, \theta)$ and $\text{FNR}(\mathcal{X}_{\text{test}}, \text{score}, \theta) = 1 - \text{TPR}(\mathcal{X}_{\text{test}}, \text{score}, \theta)$. Thus, *false positive rate (FPR)* and *true negative rate (TNR)* measure one error type and *false negative rate (FNR)* and *true positive rate (TPR)* measure the other error type. For directly comparing two different anomaly detection systems, a single metric incorporating both error types is much easier to handle. The most commonly used metric for this purpose is the F_1 score given by

$$\begin{aligned} F_1(\mathcal{X}_{\text{test}}, \text{score}, \theta) & \\ & := \frac{\text{TP}(\mathcal{X}_{\text{test}}, \text{score}, \theta)}{\text{TP}(\mathcal{X}_{\text{test}}, \text{score}, \theta) + 0.5(\text{FP}(\mathcal{X}_{\text{test}}, \text{score}, \theta) + \text{FN}(\mathcal{X}_{\text{test}}, \text{score}, \theta))} \quad (20) \\ & = \frac{2 \cdot \text{TP}(\mathcal{X}_{\text{test}}, \text{score}, \theta)}{|\mathcal{A}_{\text{pred}}(\mathcal{X}_{\text{test}}, \text{score}, \theta)| + |\mathcal{X}_{\text{test}} \cap \mathcal{X}_{\text{anomalous}}|} \in [0, 1], \end{aligned}$$

which is the harmonic mean of *precision* and *recall* defined as

$$\text{precision}(\mathcal{X}_{\text{test}}, \text{score}, \theta) := \frac{\text{TP}(\mathcal{X}_{\text{test}}, \text{score}, \theta)}{|\mathcal{A}_{\text{pred}}(\mathcal{X}_{\text{test}}, \text{score}, \theta)|} \in [0, 1] \quad (21)$$

and

$$\text{recall}(\mathcal{X}_{\text{test}}, \text{score}, \theta) := \text{TPR}(\mathcal{X}_{\text{test}}, \text{score}, \theta) \in [0, 1]. \quad (22)$$

Determining an optimal decision threshold in a semi-supervised setting is highly non-trivial (see [Section 2.11](#)) and threshold-dependent evaluation metrics largely depend on the quality of the estimated decision threshold. Hence, threshold-dependent evaluation metrics may not accurately reflect the performance of an ASD system.

Threshold-independent evaluation metrics are more objective since they do not rely on a chosen threshold and thus give a more complete picture of the performance of an anomaly detection system [3, 48]. Nevertheless, for any practical application determining a decision threshold is still needed and thus the F_1 score is still an important evaluation metric despite being less suitable for comparing system performances than threshold-independent metrics. A commonly used approach is to utilize both error types, false positives and false negatives, for all possible thresholds and plot the results against each other. Examples are the receiver operating characteristic (ROC)-curve, which uses [FPR](#) and [TPR](#), and the detection error tradeoff (DET)-curve, which uses [FPR](#) and [FNR](#) as depicted in [Figure 8](#). Since the curves themselves are difficult to compare to each other for different anomaly detection systems, metrics expressed as a single number are derived from these curves. One possibility is to take the *area under the receiver operating characteristic curve (AUC-ROC)* [19], which is also the most commonly used metric for evaluating ASD systems.

Let $(\theta_{\text{sorted}}(\mathbf{k}))_{\mathbf{k}=1, \dots, |\mathcal{X}_{\text{test}}|} \subset \mathbb{R}$ denote a monotonically increasing sequence of threshold values. Calculating the [AUC-ROC](#) with linearly spaced thresholds requires an infinitesimally fine resolution to yield exact results. Because of this,

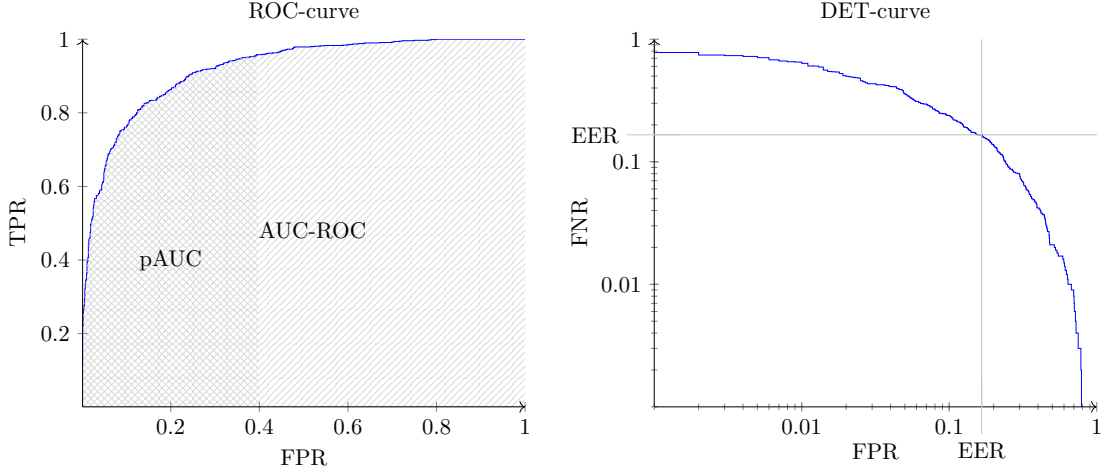


Figure 8: Examples of [ROC](#)- and [DET](#)-curves including the metrics [AUC-ROC](#), [pAUC](#) (with $p = 0.4$) and [EER](#). For [DET](#)-curves, usually a logarithmic scaling of the axis is used.

only the threshold values for which the intermediate statistics, i.e. [FPR](#) and [TPR](#), change are used as this leads to improved computational efficiency and higher accuracy than linearly spaced thresholds [48]. Formally, this corresponds to setting $\theta_{\text{sorted}}(k) := \text{sort}(\text{score}(X_{\text{test}}))(k)$ where [sort](#) denotes a function sorting real values from low to high and $\text{score}(X_{\text{test}}) := \{\text{score}(x) \in \mathbb{R} : x \in X_{\text{test}}\} \subset \mathbb{R}$. Using the trapezoidal rule, the [AUC-ROC](#) score can be approximated by

$$\begin{aligned} & \text{AUC-ROC}(X_{\text{test}}, \text{score}) \\ & \approx \sum_{k=1}^{|X_{\text{test}}|-1} \Delta_{\text{mean}} \text{TPR}(X_{\text{test}}, \text{score}, k) \cdot \Delta_{\text{diff}} \text{FPR}(X_{\text{test}}, \text{score}, k) \end{aligned} \quad (23)$$

where

$$\begin{aligned} & \Delta_{\text{diff}} \text{FPR}(X_{\text{test}}, \text{score}, k) \\ & := \text{FPR}(X_{\text{test}}, \text{score}, \theta_{\text{sorted}}(k+1)) - \text{FPR}(X_{\text{test}}, \text{score}, \theta_{\text{sorted}}(k)) \end{aligned} \quad (24)$$

and

$$\begin{aligned} & \Delta_{\text{mean}} \text{TPR}(X_{\text{test}}, \text{score}, k) \\ & := \frac{\text{TPR}(X_{\text{test}}, \text{score}, \theta_{\text{sorted}}(k+1)) + \text{TPR}(X_{\text{test}}, \text{score}, \theta_{\text{sorted}}(k))}{2}. \end{aligned} \quad (25)$$

For all experiments conducted in this thesis, the implementation provided in [scikit-learn](#) [176], which is based on the trapezoidal rule, was used. A more intuitive interpretation of the [AUC-ROC](#) is given by the following theorem [3, 71]:

Theorem 2.10. *The [AUC-ROC](#) score is equal to the probability that $\text{score}(x_n) < \text{score}(x_a)$ holds for random samples $x_n \in X_{\text{normal}} \cap X_{\text{test}}$ and $x_a \in X_{\text{anomalous}} \cap X_{\text{test}}$.*

The evaluation metric *partial area under the receiver operating characteristic curve* ($pAUC$) is the $AUC-ROC$ for a restricted part of the ROC -curve to any range of FPR [143], TPR [91] or both [266]. The idea is that some parts of the ROC -curve are often irrelevant in practice and by restricting the evaluation metric to a range of FPR or TPR relevant for a particular application the metric becomes more meaningful. For anomaly detection, this usually means to ensure a low FPR because for high FPR values users of an anomaly detection system will not take any occurring alarms, i.e. detected anomalies, seriously because too many of them are false alarms. Therefore, throughout this thesis $pAUC$ will denote the $AUC-ROC$ under low FPR ranging from 0 to a parameter $p \in (0, 1]$ to be specified.

In biometric applications such as speaker verification, often the *equal error rate* (EER) is used as an alternative threshold-independent evaluation metric. The EER is the point on the DET -curve where FPR and FNR are equal to each other. Due to the inverse relation of FNR and TPR given by $FNR(X_{\text{test}}, \text{score}, \theta) = 1 - TPR(X_{\text{test}}, \text{score}, \theta)$, the EER can also be easily determined using the ROC -curve.

2.10.2 Open-set classification

OSC can be decomposed into two subtasks: Anomaly detection and closed-set classification. Hence, there are not only false positives and false negatives as possible errors but also a so-called *confusion error* corresponding to true positive samples for which an incorrect class is predicted. One can evaluate multiple evaluation metrics for both subtasks, i.e. anomaly detection and CSC individually or combine them into a single metric. Let $(X_{\text{normal}}^{(i)})_{i=1, \dots, N_{\text{classes}}}$ be a partition of X_{normal} where $X_{\text{normal}}^{(i)} \subset X_{\text{normal}}$ for $i = 1, \dots, N_{\text{classes}}$ denotes the i -th known class. Let $\text{pred}_{\text{class}} : X \rightarrow \{1, \dots, N_{\text{classes}}\}$ denote a classifier and

$$C_{\text{pred}}^{(i)}(X_{\text{test}}, \text{pred}_{\text{class}}) := \{x \in X_{\text{test}} : \text{pred}_{\text{class}}(x) = i\} \subset X_{\text{test}} \quad (26)$$

denote all samples classified as belonging to class $i \in \{1, \dots, N_{\text{classes}}\}$. A simple way to combine two metrics, one for each subtask, is to use a weighted sum of the accuracy obtained when classifying between the known classes and the accuracy obtained for anomaly detection viewed as a binary classification task [149]:

$$\begin{aligned} & \text{ACC}(X_{\text{test}}, \text{score}, \theta, \text{pred}_{\text{class}}, X_{\text{normal}}^{(1)}, \dots, X_{\text{normal}}^{(N_{\text{classes}})}, \beta_{\text{acc}}) \\ & := \beta_{\text{acc}} \sum_{i=1}^{N_{\text{classes}}} \frac{|C_{\text{pred}}^{(i)}(X_{\text{test}}, \text{pred}_{\text{class}}) \cap X_{\text{normal}}^{(i)}|}{|X_{\text{normal}}^{(i)} \cap X_{\text{test}}|} \\ & \quad + (1 - \beta_{\text{acc}}) \frac{\text{TP}(X_{\text{test}}, \text{score}, \theta) + \text{TN}(X_{\text{test}}, \text{score}, \theta)}{|X_{\text{test}}|} \in [0, 1] \end{aligned} \quad (27)$$

with $\beta_{\text{acc}} \in [0, 1]$. A common choice is to set $\beta_{\text{acc}} = 0.5$ since, without prior knowledge, both subtasks are equally important.

Threshold-independent evaluation metrics for **OSC** can be defined similarly to the ones defined for anomaly detection. In fact, the metric $\text{top-}N_{\text{classes}}$ **EER** [205] is the same as **EER** for anomaly detection. The idea of using this metric is that the lowest anomaly score should be obtained for any of the known classes and not for one of the unknown classes. However, this evaluation metric ignores the **CSC** subtask of an **OSC** problem and thus has only limited meaning. Another more meaningful metric, also incorporating the **CSC** subtask, is the $\text{top-}1$ **EER** [205]. For this evaluation metric, **FNR** is the same as before but the **FPR** is different because confused classes are another error type for normal samples and are also treated as false positives. Formally, define

$$\begin{aligned} & \text{FP}_{\text{top-}1}(\mathbf{X}_{\text{test}}, \text{score}, \boldsymbol{\theta}, \text{pred}_{\text{class}}, \mathbf{X}_{\text{normal}}^{(1)}, \dots, \mathbf{X}_{\text{normal}}^{(N_{\text{classes}})}) \\ & := \text{FP}(\mathbf{X}_{\text{test}}, \text{score}, \boldsymbol{\theta}) \\ & \quad + \sum_{i=1}^{N_{\text{classes}}} |\mathbf{X}_{\text{test}} \setminus (\mathcal{A}_{\text{pred}}(\mathbf{X}_{\text{test}}, \text{score}, \boldsymbol{\theta}) \cup \mathcal{C}_{\text{pred}}^{(i)}(\mathbf{X}_{\text{test}}, \text{pred}_{\text{class}})) \cap \mathbf{X}_{\text{normal}}^{(i)}| \end{aligned} \quad (28)$$

as well as

$$\begin{aligned} & \text{FPR}_{\text{top-}1}(\mathbf{X}_{\text{test}}, \text{score}, \boldsymbol{\theta}, \text{pred}_{\text{class}}, \mathbf{X}_{\text{normal}}^{(1)}, \dots, \mathbf{X}_{\text{normal}}^{(N_{\text{classes}})}) \\ & := \frac{\text{FP}_{\text{top-}1}(\mathbf{X}_{\text{test}}, \text{score}, \boldsymbol{\theta}, \text{pred}_{\text{class}}, \mathbf{X}_{\text{normal}}^{(1)}, \dots, \mathbf{X}_{\text{normal}}^{(N_{\text{classes}})})}{|\mathbf{X}_{\text{test}} \cap \mathbf{X}_{\text{normal}}|} \in [0, 1]. \end{aligned} \quad (29)$$

Using these definitions, one can determine the $\text{top-}1$ **EER** as the point where $\text{top-}1$ **FPR** and **FNR** are equal. Similarly, the $\text{top-}1$ **AUC-ROC** and $\text{top-}1$ **pAUC** can be defined by replacing the regular **FPR** with the $\text{top-}1$ **FPR** for the **ROC**-curve. Hence, all threshold-independent metrics used for anomaly detection have been generalized for **OSC**.

2.10.3 Sound event detection

When detecting sound events, not only the correct sound event class but also the temporal position of a detected event has to be recognized. Therefore, specifically designed evaluation metrics have to be used that also take the temporal position of a detected sound event into account. In this section, we will mainly follow [148] and [48] to give an overview of evaluation metrics for **SED**, which in most cases are threshold-dependent evaluation metrics.

First, one can distinguish between *segment-based*, *event-based* (also called *collar-based*) and *intersection-based* [16, 51] evaluation metrics [48] for **SED**. Segment-based metrics divide the audio signal into temporal segments of fixed size and each annotated event that is contained in a given segment needs to be detected by the **SED** system. However, this causes long events ranging over multiple segments to have a higher contribution to the score. Furthermore, interruptions of detections belonging to single events may occur unnoticed. Event-based evaluation metrics

solve these issues by comparing on- and offsets of annotated and detected events and checking whether the differences are below a pre-defined value, called *collar*. If they are below this collar, the corresponding detections are regarded as errors. But for these metrics, ambiguous definitions of sound events that can either be labeled or detected as a single long event or multiple short events have a strong impact on the performance. Intersection based metrics solve these issues by comparing the size of the intersection of a detected event and the corresponding annotation of the same event, normalized by the total length of the annotated event, to a pre-defined threshold [51].

Another distinction to be made is between *micro-averaged* or *macro-averaged* evaluation metrics, which are also being referred to as *instance-based averaging* and *class-based averaging*, respectively [148]. For micro-averaging, each individual segment or sound event has the same influence on the evaluation metric. This is achieved by evaluating each segment or sound event independently. For macro-averaging, each class has the same influence on the evaluation metric. This is achieved by computing class-wise evaluation metrics and taking their mean. Micro-averaged metrics emphasize classes for which more samples are evaluated whereas macro-averaged metrics put more weight on samples belonging to classes with fewer test samples. Hence, both strategies handle imbalanced classes differently.

One possibility to calculate an evaluation metric for **SED** is to determine intermediate statistics such as **TPs** and **FPs** for each segment or sound event. These statistics can be combined by applying micro- or macro-averaging and then an evaluation metric such as the F_1 score can be computed. Another commonly used evaluation metric is the *error rate* that is based on three types of errors: *Substitutions*, *deletions* and *insertions*. Substitutions are identified as sound events that are mistakenly recognized as belonging to a different class than they actually do, similar to the confusion error. Deletions are annotated sound events that have not been detected (false negatives) and insertions are detections for which no corresponding sound event has been annotated (false positives). Using these three error types, the error rate is the sum over all errors divided by the total number of annotated instances to be detected. Note that macro-averaging does not consider substitutions as each class is processed individually and thus only a micro-averaged error rate can be computed.

Although, threshold-dependent evaluation metrics are usually used for **SED**, threshold-independent metrics can also be defined by generalizing the threshold-independent metrics for **OSC** but are still subject to active research [16, 48, 51]. In [16], the main idea is to compute **ROC**-curves for each sound event class and summarize these class-wise curves into a single curve called polyphonic sound detection (**PSD**)-**ROC** to compute a threshold-independent metric. Due to the large number of possible thresholds, these curves are often approximated using a finite set of thresholds, which may result in underestimating the true evaluation metric. [48] solves this computational issue by jointly monitoring changes of the intermediate statistics as for example **TP** and **FP** caused by changing the decision

thresholds for all sound event classes and computing the final evaluation metrics from the cumulative sums of these statistics.

2.11 DECISION THRESHOLD ESTIMATION

Regardless of whether threshold-dependent or threshold-independent evaluation metrics are used when developing an ASD system, for practical applications a decision threshold is needed to decide whether a given sample is normal or anomalous. For semi-supervised ASD, i.e. without access to anomalous training samples, it is impossible to directly optimize a decision threshold through evaluation and the decision threshold needs to be estimated using normal data only. Experienced users of an ASD system can also manually adjust the decision threshold based on their expertise. But even then, a well-tuned default setting of the threshold is still helpful. Moreover, their experience is based on observing real anomalies and access to these anomalous samples would also allow to automatically set the decision threshold by optimizing a threshold-dependent evaluation metric. In a semi-supervised setting, the general idea of most estimation methods boils down to finding a decision threshold that separates the most extreme values, e.g. the largest 10 percent, of the anomaly scores belonging to normal samples from the remaining anomaly scores. By doing so, one hopes that this decision threshold is also a good estimate for separating the anomaly scores of normal and anomalous data. Most of these methods are based on the assumption that the anomaly scores have a specific distribution, typically a normal distribution, and that normal and anomalous samples belonging to the training and test dataset follow the same distributions. Hence, the quality of the estimated decision threshold does not depend on the estimation method alone but also on the chosen method for computing the anomaly scores.

There are many different approaches for finding decision thresholds [189, 267]. Arbitrary anomaly scores, which are potentially biased, can be calibrated by converting them into (pseudo-)probabilities for which thresholds can be determined [55, 56]. Then, a fixed probability of 0.5 can be used as a decision threshold for these calibrated scores. However, this does not solve the problem of estimating a decision threshold but only reformulates the problem to estimating a calibration function. For multivariate data or scores, a threshold can be determined by using the empirical distribution function of the squared Mahalanobis distance and using a critical value such as a small quantile of the chi-squared distribution, which is the theoretical distribution function of this empirical distribution, as a threshold [195]. An extension of this approach uses an adaptive threshold [52]. However, anomaly scores used for ASD are usually univariate and thus multivariate approaches are not needed. When continuously monitoring audio data streams for anomalies, these data streams themselves consist of sound events of which most are normal and only a few are anomalous. Therefore, one can utilize previously encountered events and try to detect changes occurring in the stream of anomaly scores [31, 63, 273]. As

most publicly available datasets for academic use consist of multiple short recordings instead of a long audio stream, only estimation techniques that process the anomaly scores of each recording individually will be discussed in this thesis. Note that the 90th percentile used by most of these methods is the standard value that can be found in the literature but is not guaranteed to be optimal.

In the following, several threshold estimation techniques will be listed. Let $\text{score}(\mathbf{Y})$ denote the anomaly scores belonging to a finite number of normal training samples $\mathbf{Y} \subset \mathbf{X}_{\text{train}} \subset \mathbf{X}_{\text{normal}}$.

- *Gamma distribution percentile (GDP)* assumes that the anomaly scores follow a gamma distribution. The inverse of the 90th percentile of the empirical cumulative distribution function is used as the decision threshold. This is the most commonly used approach for ASD in machine condition monitoring [11, 46, 96, 126, 163, 182, 270].
- *Histogram percentile (HP)* directly uses the histogram of the anomaly scores without fitting a distribution first. Note that this silently assumes a uniform distribution. Again, the 90th percentile is used as the decision threshold.
- *Standard deviation (SD)* assumes the anomaly scores to be normally distributed. The decision threshold is set to

$$\theta_{\text{SD}}(\mathbf{Y}, \text{score}, \beta_{\text{SD}}) := \text{mean}(\text{score}(\mathbf{Y})) + \beta_{\text{SD}} \cdot \text{std}(\text{score}(\mathbf{Y})) \quad (30)$$

where $\text{mean}(\text{score}(\mathbf{Y}))$ and $\text{std}(\text{score}(\mathbf{Y}))$ denote the mean and standard deviation of the anomaly scores and $\beta_{\text{SD}} \in \mathbb{R}_+$ is a hyperparameter to be chosen. To have a consistent evaluation with the previous two approaches, one can use $\beta_{\text{SD}} = 1.28$, which approximately corresponds to the 90th percentile.

- *Interquartile range (IQR)* utilizes the first and third quartile of the anomaly scores, denoted by $\text{Q1}(\text{score}(\mathbf{Y})) \in \mathbb{R}$ and $\text{Q3}(\text{score}(\mathbf{Y})) \in \mathbb{R}$, respectively. This means that $\text{score}(\mathbf{y}) \geq \text{Q1}(\text{score}(\mathbf{Y}))$ for 75% of the samples $\mathbf{y} \in \mathbf{Y}$ and $\text{score}(\mathbf{y}) \geq \text{Q3}(\text{score}(\mathbf{Y}))$ for 25% of the samples $\mathbf{y} \in \mathbf{Y}$. Then,

$$\theta_{\text{IQR}}(\mathbf{Y}, \text{score}, \beta_{\text{IQR}}) := \text{Q3}(\text{score}(\mathbf{Y})) + \beta_{\text{IQR}} \cdot (\text{Q3}(\text{score}(\mathbf{Y})) - \text{Q1}(\text{score}(\mathbf{Y}))) \quad (31)$$

is used as a decision threshold. Therefore, the approach assumes a uniform distribution, similar to HP. Typically, a value of $\beta_{\text{IQR}} = 1.5$ is used [194]. This approach is also known as *boxplot* [189].

- *Mean absolute deviation (MAD)* is based on the assumption that the median is more robust against anomalies than the mean. Here, the decision threshold is set to

$$\begin{aligned} \theta_{\text{MAD}}(\mathbf{Y}, \text{score}, \beta_{\text{MAD}}, \delta_{\text{MAD}}) &:= \text{median}(\text{score}(\mathbf{Y})) \\ &+ \delta_{\text{MAD}} \cdot \beta_{\text{MAD}} \cdot \text{median}_{\mathbf{y} \in \mathbf{Y}}(|\text{score}(\mathbf{y}) - \text{median}(\text{score}(\mathbf{Y}))|), \end{aligned} \quad (32)$$

where $\text{median}(\text{score}(Y))$ denotes the median of the score values $\text{score}(Y)$ and the hyperparameter δ_{MAD} is usually set to $\delta_{\text{MAD}} = 1.4826$ [194]. For the hyperparameter β_{MAD} , [267] proposes to use $\beta_{\text{MAD}} = 3$ and [189] uses $\beta_{\text{MAD}} = 2$.

- *OCSVMs* [202] can be used to estimate the support of a distribution. This is done by discriminating between regions of high and low density using a hyperplane in a high-dimensional space. When training a *OCSVM* with anomaly scores, the learned hyperplane can be used as a decision threshold. For the hyperparameter ν_{SVM} , a value of 0.1 can be used which corresponds to treating 10% of the anomaly scores as anomalous, i.e. using the 90th percentile as done for the previously presented approaches.
- *Generalized extreme studentized deviate (GESD)* [193] is an iterative approach based on the Grubbs's test [69]. This statistical test assumes a normal distribution and is calculated on the so-called Grubbs statistic defined as

$$\text{Grubbs}(\text{score}(Y)) = \frac{|\max_{\mathbf{y} \in Y} \text{score}(\mathbf{y}) - \text{mean}(\text{score}(Y))|}{\text{std}(\text{score}(Y))} \quad (33)$$

where $\text{mean}(\text{score}(Y))$ and $\text{std}(\text{score}(Y))$ denote the mean and standard deviation of the anomaly scores, respectively. The Grubbs statistic, denoted by $\text{Grubbs}(\text{score}(Y))$, is evaluated against the upper critical value of the student's t-distribution with a significance level $\delta_{\text{Grubbs}} = 0.05$ and data size $|Y|$:

$$\text{Grubbs}(\text{score}(Y)) > \frac{|Y| - 1}{\sqrt{|Y|}} \sqrt{\frac{t_{\delta_{\text{Grubbs}}/(2|Y|), |Y|-2}^2}{|Y| - 2 + t_{\delta_{\text{Grubbs}}/(2|Y|), |Y|-2}^2}}. \quad (34)$$

Note that the Grubbs test only determines whether the highest anomaly score corresponds to an anomalous sample. For *GESD*, Grubbs is repeated by iteratively removing the highest anomaly score until the Grubbs condition is not met. The last anomaly score, which is removed by this procedure, is the resulting decision threshold.

- *CleverSD* [20] is another iterative approach. The idea is to iteratively apply *SD* to determine a decision threshold and remove the highest anomaly score from the training scores in case it is above this decision threshold. This procedure is repeated until the highest anomaly score is below the decision threshold estimated by *SD*. Again, the last anomaly score removed by this approach is the resulting decision threshold.
- *Multi-stage thresholding (MST)* [267] is yet another iterative approach that generalizes *cleverSD*. The idea is to simply apply a non-iterative method multiple times. In contrast to *cleverSD*, not only the highest score but all scores above the decision threshold are removed. In [267] it has been experimentally shown that two iterations are usually sufficient.

Note that the estimation methods listed above can also be applied for unsupervised [ASD](#) where the extreme values of the anomaly scores are likely to be truly anomalous and thus the estimated decision thresholds are also more likely to be a good estimate of the optimal threshold. In a semi-supervised setting there is no guarantee that the extreme values of the anomaly scores belonging to normal samples have a similar magnitude as anomaly scores belonging to anomalous samples. Both can be very different, which is the reason why these estimated thresholds usually do not lead to optimal results. In conclusion, to some extent it is even more difficult to estimate a decision threshold in a semi-supervised setting than it is in an unsupervised setting. However, training the embedding model is more difficult in an unsupervised setting.

2.12 SUMMARY

In this chapter, state-of-the-art audio embeddings for semi-supervised [ASD](#) and the building blocks for designing an [ASD](#) system that utilizes these embeddings have been reviewed. Such a system consists of a frontend for pre-processing the audio data, an embedding model and a backend for deciding between normal and anomalous samples. Each of these three components will now be briefly summarized.

The frontend consists of computing feature representations from the raw audio signals to be used as input for the embedding models. The main goal is to reduce the high dimension of the audio signals while not losing or even highlighting information relevant for the [ASD](#) task. This consists of three steps: 1) Pre-processing the audio signals such that they have the same sampling rate and possibly the same duration, 2) computing spectral features to reduce their dimension and 3) normalizing them to ensure that all dimensions of the input representations are scaled similarly.

Next, three different types of embeddings that are computed by further processing the input feature representations with neural networks were presented: One-class embeddings, auxiliary task embeddings and pre-trained embeddings. One-class embeddings aim at learning embedding spaces that are suitable for autoencoding the input data or learn a hypersphere of minimal volume. The major difficulty is to prevent learning trivial solutions that do not require the embeddings to contain useful information for identifying anomalous data. Auxiliary task embeddings are learned by classifying between classes provided by available meta information or by applying [SSL](#) and thus may require additional knowledge about the data. Pre-trained embeddings are obtained by training an embedding model on a large dataset that does not need to be related to the target application assuming that the learned embeddings also contain useful information for the target application, which may not always be a valid assumption.

Using one of the embeddings, different ways of computing an anomaly score were presented. These mainly consist of estimating the distribution of embeddings

belonging to normal samples and can be as simple as calculating the Euclidean distance to the mean of the embeddings. To train an embedding model, multiple data augmentation techniques that can be used to improve the ASD performance were examined, namely mixup, SpecAugment and simulating anomalous samples for training by modifying normal training samples. Furthermore, the three different embedding types were compared to each other with respect to multiple criteria such as performance, computational requirements and usability showing that each embedding type has advantages and disadvantages over the other two types and none is superior for every possible application. In addition, ensembling methods used to combine multiple models for boosting the ASD performance such as using a weighted sum of anomaly scores were listed, which may be used to combine the strengths of several approaches.

To measure the performance of a system and be able to compare multiple systems, evaluation metrics for ASD, OSC and SED were presented in Section 2.10. Mainly, one can differentiate between threshold-dependent performance measures that depend on a specifically chosen decision threshold applied to the anomaly scores, and threshold-independent performance measures. Estimating a decision threshold for semi-supervised ASD is a difficult task and requires sophisticated techniques. The reason is that only anomaly scores belonging to normal samples are available. Thus, one needs to assume that the extreme values of these anomaly scores, e.g. all scores above the 90th percentile have a similar magnitude as the anomaly scores belonging to anomalous samples. Then, the value separating the extreme values from the rest can be used as a decision threshold.

As seen above, semi-supervised ASD is an active research area providing many different choices when designing a system for a particular application. The goal of the next chapter is to identify approaches that perform particularly well by comparing several of the methods presented above as well as novel methods and design an ASD system based on these findings.

In the previous chapter, several possibilities to design an ASD system were presented. The goal of this chapter is to design an ASD system based on audio embeddings that yields state-of-the-art performance. To this end, several design choices will be compared. As the main difficulty for semi-supervised anomaly detection is to be able to train a model without access to anomalous data, the focus will be on choosing a suitable loss function for training the embedding model and how to calculate anomaly scores.

This chapter is structured as follows: First, the experimental setup consisting of an ASD dataset and a baseline model, whose performance will be optimized, is presented. Then, different loss functions, namely one-class losses and angular margin losses will be compared to each other, both theoretically and experimentally. Third, the sub-cluster AdaCos loss, which is a generalization of the AdaCos loss, will be defined and investigated.

3.1 CONTRIBUTIONS OF THE AUTHOR

The sections of this chapter are largely based on the following key publications:

- Kevin Wilkinghoff. “Sub-Cluster AdaCos: Learning Representations for Anomalous Sound Detection.” In: *International Joint Conference on Neural Networks*. IEEE, 2021. DOI: [10.1109/IJCNN52387.2021.9534290](https://doi.org/10.1109/IJCNN52387.2021.9534290).
- Kevin Wilkinghoff and Frank Kurth. “Why do Angular Margin Losses work well for Semi-Supervised Anomalous Sound Detection?” In: *IEEE/ACM Transactions on Audio, Speech and Language Processing* 32 (2024), pp. 608–622. DOI: [10.1109/TASLP.2023.3337153](https://doi.org/10.1109/TASLP.2023.3337153).

For publications that are not single-authored, individual contributions of the thesis author and all co-authors to these publications are stated in [Section A.1](#). If not stated otherwise, the content listed in the following paragraph is the sole contribution of the thesis author.

[Section 3.2.2](#) and [Sections 3.4.3 to 3.4.7](#) are based on [\[242\]](#). [Sections 3.3, 3.4.1 and 3.4.2](#). The experimental results shown in [Figure 9](#) and [Table 5](#) are adapted from [\[262\]](#) by changing the dataset used for the experiments.

3.2 EXAMPLE APPLICATION: MACHINE CONDITION MONITORING

As already announced in [Section 1.3](#), acoustic machine condition monitoring will serve as the main example application for semi-supervised ASD in this thesis.

The main reason for this choice is that most recently published works on semi-supervised [ASD](#) utilize publicly available datasets for acoustic machine condition monitoring that belong to the annual DCASE Challenge. Also utilizing these datasets ensures that the results are reproducible and there is a commonly used experimental setup in the community to compare findings experimentally. Another advantage is that the task is well-defined because normal recordings belong to fully functioning machines and all anomalies indicate mechanical failure caused on purpose in a controlled environment. Last but not least, the dataset provides a difficult task because real factory background noise was added to all recordings, which fosters research for semi-supervised [ASD](#).

3.2.1 *Experimental setup*

For the experiments conducted in this chapter, the dataset belonging to the task “Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring” of the DCASE2020 Challenge [108] is used¹. The dataset contains recordings of the machine types “fan”, “pump”, “slider” and “valve” from MIMII [188] as well as “ToyCar” and “ToyConveyor” from ToyADMOS [106], which also contain real factory background noise. Each recording has a length of 10s and a sampling rate of 16 kHz. The dataset is divided into two subsets, a development set and an evaluation set. Each of these sets consists of a training split, which only contains normal data, and a test split, which contains normal and anomalous data. There are multiple individual machines of each type. The set of all recordings belonging to a specific machine ID is called section. During testing, the section a given recording belongs to is known and can also be used as input to the [ASD](#) system. The sections contained in the development set and evaluation set are mutually exclusive but are the same for the training and test splits. In total, there are 41 different sections. Further details about the dataset structure can be found in [Table 3](#).

To evaluate the performance of an [ASD](#) system, the [AUC-ROC](#) and [pAUC](#) with $p = 0.1$ are determined for each section of the dataset. To obtain a single value that can be used as a performance measure to rank the systems, the arithmetic mean of all [AUC-ROCs](#) and [pAUCs](#) over all sections is calculated.

3.2.2 *Baseline model for extracting embeddings*

As stated in [Section 2.4](#), training a model that uses an auxiliary classification task is a commonly used approach for [ASD](#) that attains state-of-the-art performance. Because of this, such an embedding model is used as a baseline model, whose performance serves as a basis for measuring improvement. The structure of the baseline model is shown in [Table 4](#). It is based on a modified ResNet architecture

¹ Although the name of the task implies an unsupervised [ASD](#) setting, it is actually semi-supervised.

Table 3: Structure of the DCASE2020 [ASD](#) dataset containing recordings of 6 machine types. The set containing all recordings belonging to one individual machine is called a section.

subset	number of sections (per machine type)	split	number of recordings (per section)	
			normal	anomalous
development set	3 – 4	training	~ 1000	0
		test	100 – 200	100 – 200
evaluation set	3 – 4	training	~ 1000	0
		test	~ 200	~ 200

Table 4: Modified ResNet architecture used as the baseline model. © 2021 IEEE

layer name	structure	output size
input	-	313×128
2D convolution	7×7 , stride= 2	$157 \times 64 \times 16$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$78 \times 31 \times 16$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$39 \times 16 \times 32$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$20 \times 8 \times 64$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$10 \times 4 \times 128$
max pooling	10×1 , stride= 1	$1 \times 4 \times 128$
flatten	-	512
dense (embedding)	no activation	128

[77] mainly consisting of four residual blocks. In each convolutional layer, batch normalization [86] is applied and leaky ReLU [139] with a slope coefficient of 0.1 is used as an activation function. As input feature representations, log-Mel spectrograms with 128 Mel bins, a window size of 1024 and a hop size of 512 are computed using a high-pass filter with a cutoff frequency of 200 Hz. These features are standardized by subtracting the temporal mean and dividing with the temporal standard deviation computed by using all normal training samples. To train this embedding model, the angular margin loss AdaCos [274] is minimized by using the sections of the DCASE2020 ASD dataset corresponding to different machine IDs as classes and applying mixup [272]. More concretely, the model is trained for 400 epochs using Adam [101] with a batch size of 64. Individual design choices for improving the performance of this model will be investigated in the following sections. The minimum cosine distance of a test sample to all centers of the AdaCos loss is used to compute an anomaly score. Since only threshold-independent evaluation metrics are used to compute the performance, a decision threshold does not need to be estimated. Before evaluating the performance of this baseline model, it will be shown that using the angular margin loss AdaCos to train the model is a reasonable choice.

3.3 RELATION BETWEEN ONE-CLASS AND ANGULAR MARGIN LOSSES

When designing an ASD system based on audio embeddings, one of the major choices to make is to decide on how to train the embedding model. Compared to directly trained models, pre-trained models are known to have inferior performance in case enough training data is available because pre-trained models are less specialized. Therefore, one needs to decide between a one-class loss or an angular margin loss, which are the other two commonly used loss functions for training an embedding model, to obtain state-of-the-art performance. In this section, the relation between the compactness loss and AdaCos as representatives of one-class losses and angular margin losses, respectively, will be investigated. Before doing this, it will be shown that projecting the embeddings onto the unit sphere when using a compactness loss has several advantages.

3.3.1 Compactness loss on the unit sphere

The following lemma shows that the squared Euclidean distance and the cosine similarity are closely related on the unit sphere.

Lemma 3.1. *For $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{R}^D$ with $\|\mathbf{e}_1\|_2 = \|\mathbf{e}_2\|_2 = 1$, it holds that*

$$\text{sim}(\mathbf{e}_1, \mathbf{e}_2) = 1 - \frac{\|\mathbf{e}_1 - \mathbf{e}_2\|_2^2}{2}. \quad (35)$$

Proof. Using only basic identities, we obtain

$$\begin{aligned}\|e_1 - e_2\|_2^2 &= \langle e_1 - e_2, e_1 - e_2 \rangle = \|e_1\|_2^2 + \|e_2\|_2^2 - 2\langle e_1, e_2 \rangle \\ &= 2\left(1 - \frac{\langle e_1, e_2 \rangle}{\|e_1\|_2 \|e_2\|_2}\right) = 2(1 - \text{sim}(e_1, e_2)),\end{aligned}$$

which finishes the proof. \square

Hence, the Euclidean distance and the cosine distance, which in this case is equal to the standard scalar product, are equivalent for computing an anomaly score for embeddings on the unit sphere. Projecting embeddings onto the unit sphere can be easily accomplished by dividing an embedding with its Euclidean norm because for all $0 \neq e \in \mathbb{R}^D$ it holds that

$$\left\| \frac{e}{\|e\|_2} \right\|_2 = \frac{\|e\|_2}{\|e\|_2} = 1.$$

The following definition captures this simple fact by introducing additional notation.

Definition 3.2 (Projection onto unit sphere). Let $\mathcal{S}^{\mathbb{D}-1} := \{e \in \mathbb{R}^{\mathbb{D}} : \|e\|_2 = 1\} \subset \mathbb{R}^{\mathbb{D}}$ denote the \mathbb{D} -sphere, in the following also referred to as *unit sphere*. Then

$$\begin{aligned}P_{\mathcal{S}^{\mathbb{D}-1}} : \mathbb{R}^{\mathbb{D}} &\rightarrow \mathcal{S}^{\mathbb{D}-1} \\ P_{\mathcal{S}^{\mathbb{D}-1}}(e) &:= \frac{e}{\|e\|_2}\end{aligned}\tag{36}$$

is the *projection onto the \mathbb{D} -sphere*.

Restricting the embedding space to the unit sphere essentially reduces the embedding dimension by 1 as evident by using stereographic projection. Since the dimension of the embedding space is just a hyperparameter to be chosen and, in most cases, is higher than it needs to be, the resulting performance does not degrade. On the contrary, projecting the embeddings onto the unit sphere when using the compactness loss has several advantages. First of all, doing so prevents that the network may learn $\mathbf{0} \in \mathbb{R}^{\mathbb{D}}$ as a trivial solution in case only linear operators, such as matrix multiplications or convolutions, and activation functions with $\mathbf{0}$ as a fixed point are used. Furthermore, normalizing the embedding stabilizes the training by preventing numerical issues similar to batch normalization [86]. Another advantage is that the initialization of the (non-trainable) center is simplified since two random elements of the unit sphere have equal distance with very high probability if the dimension is sufficiently high. Therefore, the center can be initialized randomly.

3.3.2 Relation between the compactness loss and AdaCos

The relation between the compactness loss and AdaCos is characterized by the following corollary.

Corollary 3.3. For $\text{Im}(\phi) \subseteq \mathcal{S}^{\mathbb{D}-1}$, minimizing $\mathcal{L}_{ada}(\mathcal{Y}, \mathcal{C}, \phi, \mathbf{w}, \text{lab})$ with gradient descent is equivalent to minimizing

$$-\frac{\tilde{s}}{2} \frac{1}{|\mathcal{Y}|} \sum_{\mathbf{x} \in \mathcal{Y}} \sum_{i=1}^{N_{classes}} \text{lab}(\mathbf{x})_i \sum_{k=1}^{N_{classes}} \text{softmax}(\hat{\mathbf{s}} \cdot \text{sim}(\phi(\mathbf{x}, \mathbf{w}), \mathbf{c}_k)) \cdot \left(\frac{\partial}{\partial \mathbf{w}} \|\phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_i\|_2^2 - \frac{\partial}{\partial \mathbf{w}} \|\phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_k\|_2^2 \right). \quad (37)$$

Proof. The proof of this corollary will be postponed to [Section 3.4](#) because the statement directly follows from a more general theorem. \square

This corollary shows that minimizing the AdaCos loss increases intra-class similarity, i.e. reduces the compactness loss for each class, while at the same time decreasing inter-class similarity. Hence, AdaCos can be seen as a multi-class version of the compactness loss that explicitly ensures a margin between classes. Note that a constant function can only be a solution for a single class as a classifier requires different solutions for different classes. Therefore, using multiple normal classes prevents that the model learns a constant function as a trivial solution. In the case of a classification task, a trivial solution corresponds to learning a perfect classifier that maps each point exactly to the center of the corresponding class and is practically impossible to obtain for non-trivial classification tasks.

As already mentioned in [Section 2.3](#), a similar strategy is employed in [\[85, 177\]](#) where an auxiliary classification task on a second dataset is used as a so-called *descriptiveness loss*. Compared to an angular margin loss, there are several differences even when the same dataset is used for both tasks. First and foremost, for angular margin losses the inter-class compactness loss is increased instead of only using a discriminative objective with a categorical crossentropy (CXE). Furthermore, a margin between different classes is explicitly ensured, which is not the case for the descriptiveness loss. Third, an angular margin loss uses a different weight for individual classes by utilizing softmax probabilities as shown in [Corollary 3.3](#).

To experimentally verify the relation between the different loss functions, the development of different loss functions over time is depicted in [Figure 9](#). It can be seen that training a model by minimizing the AdaCos loss indeed minimizes the intra-class compactness losses while maximizing the inter-class compactness losses. Note that, according to [Lemma 3.1](#), a mean squared Euclidean distance equal to 2, as attained by the inter-class loss, corresponds to an angle equal to $\frac{\pi}{2}$, i.e. orthogonality. This is an expected behavior in the embedding space because two random elements of a relatively high-dimensional vector space are orthogonal with very high probability [\[65\]](#). Furthermore, a smaller loss does not mean that

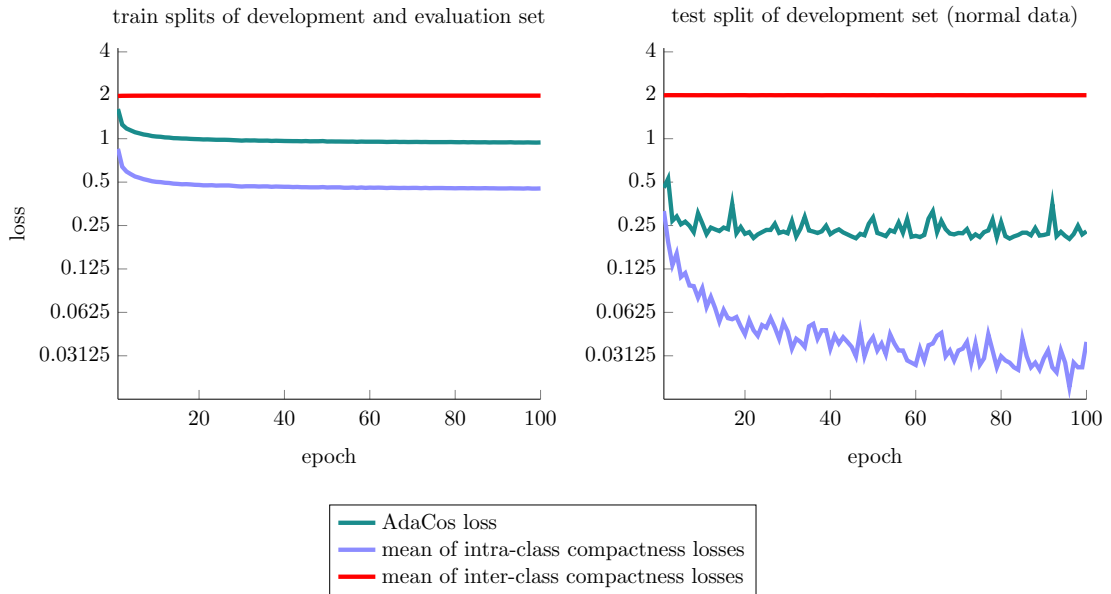


Figure 9: Temporal development of different losses obtained on the DCASE2020 dataset when training by minimizing the AdaCos loss.

the resulting [ASD](#) performance is better because minimizing any of these losses is only an auxiliary task not directly related to the [ASD](#) task.

3.3.3 Performance evaluation

In this section, the [ASD](#) performances obtained with different loss functions and using a different number of classes for an auxiliary classification task are compared. The results can be found in [Table 5](#). It can be seen that the performance improves when increasing the number of classes used for training (see second column). Using more classes increases the difficulty of the classification task and thus more information needs to be captured by the embedding model to still be able to predict the correct class. When using only a single class for the entire dataset, the performance is the same as random guessing. The most likely reason is the strong and highly diverse background noise that consists of several other sound sources and thus drowns out the anomalous signal components, which are very subtle in comparison. In contrast, using an auxiliary classification task teaches the model to closely monitor frequency bands or temporal patterns that are characteristic for the target machine sounds. This enables the embedding model to ignore the background noise whereas a one-class model does not know the difference between noise and target machine sounds and thus considers all signal components as equally important. Using the pre-defined sections of the dataset as sub-classes improves the overall performance, even when training another model for each sub-class. This indicates that individual machines, represented by the sections of the dataset, have a very distinct acoustic pattern and thus the only variability between

Table 5: Arithmetic means of all **AUC-ROCs** and **pAUCs** obtained with different losses using different auxiliary tasks over all sections of the DCASE2020 dataset. For intra-class compactness losses, non-trainable class centers and no bias terms are used to avoid learning trivial solutions. Best results in each column are highlighted with bold letters.

DCASE2020 development set			
loss	classes of auxiliary task (number of classes)	AUC	pAUC
intra-class compactness loss	none (1)	49.78 ± 0.66%	50.36 ± 0.28%
intra-class compactness loss	machine types (7)	70.61 ± 1.89%	62.58 ± 1.49%
intra-class compactness loss	machine types and sections, models trained individually (1)	67.68 ± 0.54%	57.57 ± 0.35%
intra-class compactness loss	machine types and sections (41)	89.67 ± 0.58%	83.20 ± 0.88%
intra-class compactness loss + CXE	machine types and sections (41)	90.12 ± 0.35%	83.86 ± 0.60%
AdaCos loss	machine types and sections (41)	89.30 ± 0.57%	83.03 ± 0.52%
DCASE2020 evaluation set			
loss	classes of auxiliary task (number of classes)	AUC	pAUC
intra-class compactness loss	none (1)	50.08 ± 1.07%	50.39 ± 0.59%
intra-class compactness loss	machine types (7)	73.74 ± 2.35%	65.57 ± 1.92%
intra-class compactness loss	machine types and sections, models trained individually (1)	70.60 ± 0.68%	61.10 ± 0.71%
intra-class compactness loss	machine types and sections (41)	90.55 ± 1.07%	83.54 ± 1.00%
intra-class compactness loss + CXE	machine types and sections (41)	90.84 ± 0.43%	83.85 ± 0.37%
AdaCos loss	machine types and sections (41)	90.54 ± 0.09%	84.09 ± 0.68%

different recordings of the same machine is the background noise. Still, training a joint embedding model leads to much better results than using individual models. Moreover, using an explicit classification task with a **CXE** slightly improves the performance over only increasing the intra-class similarity or using an angular margin loss. The most likely reason is that the classification task for this dataset is relatively simple due to the low variability between different recordings of the same machine, which leads to almost perfect classification results even when only using an intra-class compactness loss.

In the previous discussion of the results it was silently assumed that the background noise is similar for each class. However, if the background noise is specific for some or even all classes, then using a classification task will probably not help to improve the performance because the noise contains useful information for discriminating between the classes as well. Hence, assuming that the background noise is not class-specific is essential. For realistic applications, this is often a valid assumption because one would expect that at least some machines of different type or with a different ID are running in the same factory and share the same acoustic environment. Otherwise, defining additional sub-classes, e.g. by also providing parameter settings of machines as meta information as done in [Chapter 5](#), may be beneficial. In any case, using an angular margin loss also decreases the intra-class compactness losses and thus will not perform worse than when only using an intra-class compactness loss.

3.4 SUB-CLUSTER ADACOS

Using an angular margin loss to learn distributions of normal data enforces a Gaussian distribution for each class because the mean squared error, which is strongly related to the cosine distance (cf. Lemma 3.1), is proportional to the negative log-likelihood of a Gaussian distribution. As Gaussian distributions are relatively simple, this may be a choice that is too restrictive to accurately model the true distribution and thus may not be optimal to distinguish between normal and anomalous samples. The fact that using a CXE as a descriptiveness loss led to slightly better performance in the experiments conducted in Section 3.3.3 provides additional evidence for this claim. The goal of this section is to relax these restrictions by utilizing multiple Gaussians for each class instead of a single one and investigate the impact on the resulting ASD performance. Introducing sub-classes enables the model to learn more sophisticated distributions for each class.

The idea of allowing multiple sub-classes for each class is also used for discriminant analysis [29, 278] and was shown to outperform standard approaches such as LDA. A similar approach for angular margin losses is sub-center ArcFace [37] that has been proposed to handle noisy class labels. Apart from the non-adaptive scale parameter, the main difference to sub-cluster AdaCos is that only the closest sub-center is considered by using the minimum over all cosine-similarities. For sub-cluster AdaCos, the sum of all softmax probabilities belonging to a single class is computed during training. Taking the sum still enables the model to handle noisy class labels by assigning them to other sub-classes. Furthermore, summing the probabilities has the advantage that it is continuously differentiable whereas using the minimum distance is not.

3.4.1 Definition

The following formal definition of the sub-cluster AdaCos loss is similar to the definition of the AdaCos loss (cf. Definition 2.8) but uses $N_{\text{centers}} \in \mathbb{N}$ centers instead of a single center for each class. Furthermore, the term $B_{\text{avg}}^{(t)}$ includes the distances to all samples and not just the samples belonging to non-target classes. The reason is that sub-cluster AdaCos is intended to be used with mixup. In this, mixed up samples should be treated as anomalous samples. This modification results in a larger value for $B_{\text{avg}}^{(t)}$ and thus also a larger scale parameter leading to sharper boundaries around the distributions of the classes (cf. Figure 6).

Definition 3.4 (Sub-cluster AdaCos). Let $C_j \in \mathcal{P}(\mathbb{R}^D)$ with $|C_j| = N_{\text{centers}} \in \mathbb{N}$ denote all centers belonging to class $j \in \{1, \dots, N_{\text{classes}}\}$. Let the *dynamically adaptive scale parameter* $\hat{s}^{(t)} \in \mathbb{R}_+$ at training step $t \in \mathbb{N}_0$ be set to

$$\hat{s}^{(t)} := \begin{cases} \sqrt{2} \cdot \log(N_{\text{classes}} \cdot N_{\text{centers}} - 1) & \text{if } t = 0 \\ \frac{\text{sim}_{\text{max}}^{(t)} + \log \hat{B}_{\text{avg}}^{(t)}}{\cos\left(\min\left(\frac{\pi}{4}, \hat{\alpha}_{\text{med}}^{(t)}\right)\right)} & \text{else} \end{cases} \quad (38)$$

with

$$\hat{\mathbf{B}}_{\text{avg}}^{(t)} := \frac{1}{N_{\text{batch}}} \sum_{\mathbf{x} \in \mathcal{Y}^{(t)}} \sum_{j=1}^{N_{\text{classes}}} \sum_{\mathbf{c} \in \mathcal{C}_j} \exp(\hat{\mathbf{s}}^{(t-1)} \text{sim}(\phi(\mathbf{x}, \mathbf{w}^{(t)}), \mathbf{c}) - \text{sim}_{\text{max}}^{(t)}) \quad (39)$$

where $\hat{\alpha}_{\text{med}}^{(t)} \in [0, \frac{\pi}{2}]$ denotes the median of all mixed-up angles

$$\begin{aligned} & \lambda \sum_{\mathbf{c}^{(1)} \in \mathcal{C}_{\text{class}(\mathbf{x}_1)}} \alpha(\phi(\text{mix}_{\mathbf{x}}(\mathbf{x}_1, \mathbf{x}_2, \lambda), \mathbf{w}), \mathbf{c}^{(1)}) \\ & + (1 - \lambda) \sum_{\mathbf{c}^{(2)} \in \mathcal{C}_{\text{class}(\mathbf{x}_2)}} \alpha(\phi(\text{mix}_{\mathbf{x}}(\mathbf{x}_1, \mathbf{x}_2, \lambda), \mathbf{w}), \mathbf{c}^{(2)}) \end{aligned}$$

with $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{Y}^{(t)}$, $\lambda \in [0, 1]$, and for $\mathbf{w}^{(t)} \in \mathcal{W}$ the stability term is given by

$$\text{sim}_{\text{max}}^{(t)} := \max_{\mathbf{x} \in \mathcal{Y}^{(t)}} \max_{j=1}^{N_{\text{classes}}} \max_{\mathbf{c} \in \mathcal{C}_j} \hat{\mathbf{s}}^{(t-1)} \cdot \text{sim}(\phi(\mathbf{x}, \mathbf{w}), \mathbf{c}). \quad (40)$$

Then, the *sub-cluster AdaCos* loss is defined as

$$\begin{aligned} \mathcal{L}_{\text{sc-ada}} &: \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{P}(\mathbb{R}^D)) \times \Phi \times \mathcal{W} \times \Lambda(N_{\text{classes}}) \rightarrow \mathbb{R}_+ \\ \mathcal{L}_{\text{sc-ada}}(\mathcal{Y}, \mathcal{C}, \phi, \mathbf{w}, \text{lab}) &:= -\frac{1}{|\mathcal{Y}|} \sum_{\mathbf{x} \in \mathcal{Y}} \sum_{j=1}^{N_{\text{classes}}} \text{lab}(\mathbf{x})_j \log(\text{softmax}(\hat{\mathbf{s}} \cdot \text{sim}(\phi(\mathbf{x}, \mathbf{w}), \mathcal{C}_j))) \end{aligned} \quad (41)$$

where $|\mathcal{C}| = N_{\text{classes}}$ and, in this case,

$$\text{softmax}(\hat{\mathbf{s}} \cdot \text{sim}(\phi(\mathbf{x}, \mathbf{w}), \mathcal{C}_j)) := \sum_{\mathbf{c}_j \in \mathcal{C}_j} \frac{\exp(\hat{\mathbf{s}} \cdot \text{sim}(\phi(\mathbf{x}, \mathbf{w}), \mathbf{c}_j))}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in \mathcal{C}_k} \exp(\hat{\mathbf{s}} \cdot \text{sim}(\phi(\mathbf{x}, \mathbf{w}), \mathbf{c}_k))} \quad (42)$$

Remark. The only reason for including the stability term $-\text{sim}_{\text{max}}^{(t)}$ when calculating $\hat{\mathbf{B}}_{\text{avg}}^{(t)}$ is to improve numerical stability by reducing the argument of the exponential function. After this computation and taking the logarithm in the computation of $\hat{\mathbf{s}}^{(t)}$, the effects of the term $-\text{sim}_{\text{max}}^{(t)}$ are reversed by adding $\text{sim}_{\text{max}}^{(t)}$.

3.4.2 Relation to the compactness loss

Similar to Corollary 3.3, the relation between the sub-cluster AdaCos loss and the compactness loss is presented in the following theorem.

Theorem 3.5. Let $Y_j := \{x \in Y : \text{lab}(x)_j = 1\}$ for $j \in \{1, \dots, N_{\text{classes}}\}$. Then, for $\text{Im}(\phi) \subseteq \mathcal{S}^{\mathbb{D}-1}$ minimizing $\mathcal{L}_{sc-ada}(Y, \mathcal{C}, \phi, w, \text{lab})$ with gradient descent minimizes all intra-class compactness losses with weighted gradients given by

$$\frac{\hat{s}}{2} \sum_{i=1}^{N_{\text{classes}}} \frac{1}{|Y_i|} \sum_{x \in Y_i} \sum_{c_i \in C_i} \mathbb{P}(\tau(\phi(x, w), \mathcal{C}) = c_i | \tau(\phi(x, w), \mathcal{C}) \in C_i) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_i\|_2^2 \quad (43)$$

while maximizing all inter-class compactness losses with weighted gradients given by

$$-\frac{\hat{s}}{2} \sum_{i=1}^{N_{\text{classes}}} \frac{1}{|Y_i|} \sum_{x \in Y_i} \sum_{k=1}^{N_{\text{classes}}} \sum_{c_k \in C_k} \mathbb{P}(\tau(\phi(x, w), \mathcal{C}) = c_k) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_k\|_2^2 \quad (44)$$

where

$$\mathbb{P}(\tau(\phi(x, w), \mathcal{C}) = c_i | \tau(\phi(x, w), \mathcal{C}) \in C_i) := \frac{\exp(\hat{s} \cdot \text{sim}(\phi(x, w), c_i))}{\sum_{c'_i \in C_i} \exp(\hat{s} \cdot \text{sim}(\phi(x, w), c'_i))} \quad (45)$$

and

$$\mathbb{P}(\tau(\phi(x, w), \mathcal{C}) = c_k) := \frac{\exp(\hat{s} \cdot \text{sim}(\phi(x, w), c_k))}{\sum_{k=1}^{N_{\text{classes}}} \sum_{c'_k \in C_k} \exp(\hat{s} \cdot \text{sim}(\phi(x, w), c'_k))} \quad (46)$$

with a cluster assignment function $\tau : \mathbb{R}^{\mathbb{D}} \times \mathcal{P}(\mathcal{P}(\mathbb{R}^{\mathbb{D}})) \rightarrow \mathbb{R}^{\mathbb{D}}$ given by

$$\tau(e, \mathcal{C}) = \arg \max_{C \in \mathcal{C}} (\arg \max_{c \in C} (\text{sim}(e, c))). \quad (47)$$

Proof. Let $x \in Y$, $\phi \in \Phi$ and $\hat{s} \in \mathbb{R}_+$ be fixed and $i \in \{1, \dots, N_{\text{classes}}\}$ such that $\text{lab}(x)_i = 1$ and $\text{lab}(x)_j = 0$ for $j \neq i$. To simplify notation, define $z(w, c) := \exp(\hat{s} \cdot \text{sim}(\phi(x, w), c))$. Note that, in each iteration $t > 0$, $\hat{s}^{(t)}$ is set to a fixed value before computing the gradient of the loss function by using the current weights $w^{(t)} \in \mathcal{W}$ of the network. Therefore, $\frac{\partial \hat{s}^{(t)}}{\partial w} = 0$ although the dynamically adaptive scale parameter depends on the weights. Using Lemma 3.1, it holds that

$$\begin{aligned} \frac{\partial}{\partial w} \log \left(\sum_{c_i \in C_i} z(w, c_i) \right) &= \frac{\sum_{c_i \in C_i} z(w, c_i) \cdot \hat{s} \cdot \frac{\partial}{\partial w} \text{sim}(\phi(x, w), c_i)}{\sum_{c'_i \in C_i} z(w, c'_i)} \\ &= -\frac{\hat{s}}{2} \sum_{c_i \in C_i} \frac{z(w, c_i) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_i\|_2^2}{\sum_{c'_i \in C_i} z(w, c'_i)} \end{aligned}$$

and similarly

$$\begin{aligned}
& \frac{\partial}{\partial \mathbf{w}} \log \left(\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in C_k} z(\mathbf{w}, \mathbf{c}_k) \right) \\
&= \frac{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in C_k} z(\mathbf{w}, \mathbf{c}_k) \cdot \hat{s} \cdot \frac{\partial}{\partial \mathbf{w}} \text{sim}(\Phi(\mathbf{x}, \mathbf{w}), \mathbf{c}_k)}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in C_k} z(\mathbf{w}, \mathbf{c}'_k)} \\
&= -\frac{\hat{s}}{2} \sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in C_k} \frac{z(\mathbf{w}, \mathbf{c}_k) \cdot \frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_k\|_2^2}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in C_k} z(\mathbf{w}, \mathbf{c}'_k)} \\
&= -\frac{\hat{s}}{2} \sum_{\mathbf{c}_i \in C_i} \frac{z(\mathbf{w}, \mathbf{c}_i) \cdot \frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_i\|_2^2}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in C_k} z(\mathbf{w}, \mathbf{c}'_k)} \\
&\quad - \frac{\hat{s}}{2} \sum_{\substack{k=1 \\ k \neq i}}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in C_k} \frac{z(\mathbf{w}, \mathbf{c}_k) \cdot \frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_k\|_2^2}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in C_k} z(\mathbf{w}, \mathbf{c}'_k)}.
\end{aligned}$$

Combining both identities yields

$$\begin{aligned}
& \frac{\partial}{\partial \mathbf{w}} \sum_{j=1}^{N_{\text{classes}}} \text{lab}_j(\mathbf{x}) \log(\text{softmax}(\hat{s} \cdot \text{sim}(\Phi(\mathbf{x}, \mathbf{w}), C_j))) \\
&= \frac{\partial}{\partial \mathbf{w}} \log \left(\sum_{\mathbf{c}_i \in C_i} \frac{z(\mathbf{w}, \mathbf{c}_i)}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in C_k} z(\mathbf{w}, \mathbf{c}_k)} \right) \\
&= \frac{\partial}{\partial \mathbf{w}} \log \left(\sum_{\mathbf{c}_i \in C_i} z(\mathbf{w}, \mathbf{c}_i) \right) - \frac{\partial}{\partial \mathbf{w}} \log \left(\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in C_k} z(\mathbf{w}, \mathbf{c}_k) \right) \\
&= -\frac{\hat{s}}{2} \sum_{\mathbf{c}_i \in C_i} \frac{z(\mathbf{w}, \mathbf{c}_i) \cdot \frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_i\|_2^2}{\sum_{\mathbf{c}'_i \in C_i} z(\mathbf{w}, \mathbf{c}'_i)} \\
&\quad + \frac{\hat{s}}{2} \sum_{\mathbf{c}_i \in C_i} \frac{z(\mathbf{w}, \mathbf{c}_i) \cdot \frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_i\|_2^2}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in C_k} z(\mathbf{w}, \mathbf{c}'_k)} \\
&\quad + \frac{\hat{s}}{2} \sum_{\substack{k=1 \\ k \neq i}}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in C_k} \frac{z(\mathbf{w}, \mathbf{c}_k) \cdot \frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_k\|_2^2}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in C_k} z(\mathbf{w}, \mathbf{c}'_k)} \\
&= -\frac{\hat{s}}{2} \left(\sum_{\mathbf{c}_i \in C_i} z(\mathbf{w}, \mathbf{c}_i) \cdot \frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_i\|_2^2 \right. \\
&\quad \cdot \left(\frac{1}{\sum_{\mathbf{c}'_i \in C_i} z(\mathbf{w}, \mathbf{c}'_i)} - \frac{1}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in C_k} z(\mathbf{w}, \mathbf{c}'_k)} \right) \\
&\quad \left. - \sum_{\substack{k=1 \\ k \neq i}}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in C_k} \frac{z(\mathbf{w}, \mathbf{c}_k) \cdot \frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_k\|_2^2}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in C_k} z(\mathbf{w}, \mathbf{c}'_k)} \right)
\end{aligned}$$

$$\begin{aligned}
&= -\frac{\hat{s}}{2} \left(\sum_{\mathbf{c}_i \in \mathbf{C}_i} z(\mathbf{w}, \mathbf{c}_i) \cdot \frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_i\|_2^2 \right. \\
&\quad \cdot \left(\sum_{\substack{k=1 \\ k \neq i}}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in \mathbf{C}_k} \frac{z(\mathbf{w}, \mathbf{c}_k)}{(\sum_{\mathbf{c}'_i \in \mathbf{C}_i} z(\mathbf{w}, \mathbf{c}'_i)) (\sum_{k=1}^N \sum_{\mathbf{c}'_k \in \mathbf{C}_k} z(\mathbf{w}, \mathbf{c}'_k))} \right) \\
&\quad - \sum_{\substack{k=1 \\ k \neq i}}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in \mathbf{C}_k} \frac{z(\mathbf{w}, \mathbf{c}_k) \cdot \frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_k\|_2^2}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in \mathbf{C}_k} z(\mathbf{w}, \mathbf{c}'_k)} \Big) \\
&= -\frac{\hat{s}}{2} \sum_{\substack{k=1 \\ k \neq i}}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in \mathbf{C}_k} \frac{z(\mathbf{w}, \mathbf{c}_k)}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in \mathbf{C}_k} z(\mathbf{w}, \mathbf{c}'_k)} \\
&\quad \cdot \left(\sum_{\mathbf{c}_i \in \mathbf{C}_i} \frac{z(\mathbf{w}, \mathbf{c}_i)}{\sum_{\mathbf{c}'_i \in \mathbf{C}_i} z(\mathbf{w}, \mathbf{c}'_i)} \cdot \frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_i\|_2^2 - \frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_k\|_2^2 \right) \\
&= -\frac{\hat{s}}{2} \sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in \mathbf{C}_k} \underbrace{\frac{z(\mathbf{w}, \mathbf{c}_k)}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in \mathbf{C}_k} z(\mathbf{w}, \mathbf{c}'_k)}}_{=P(\tau(\Phi(\mathbf{x}, \mathbf{w}), \mathcal{C})=\mathbf{c}_k)} \\
&\quad \cdot \sum_{\mathbf{c}_i \in \mathbf{C}_i} \underbrace{\frac{z(\mathbf{w}, \mathbf{c}_i)}{\sum_{\mathbf{c}'_i \in \mathbf{C}_i} z(\mathbf{w}, \mathbf{c}'_i)}}_{=P(\tau(\Phi(\mathbf{x}, \mathbf{w}))=\mathbf{c}_i | \tau(\Phi(\mathbf{x}, \mathbf{w}), \mathcal{C}) \in \mathbf{C}_i)} \\
&\quad \cdot \left(\frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_i\|_2^2 - \frac{\partial}{\partial \mathbf{w}} \|\Phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_k\|_2^2 \right)
\end{aligned}$$

where it is used that

$$\begin{aligned}
&\frac{1}{\sum_{\mathbf{c}'_i \in \mathbf{C}_i} z(\mathbf{w}, \mathbf{c}'_i)} - \frac{1}{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in \mathbf{C}_k} z(\mathbf{w}, \mathbf{c}'_k)} \\
&= \frac{\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in \mathbf{C}_k} z(\mathbf{w}, \mathbf{c}_k) - \sum_{\mathbf{c}_i \in \mathbf{C}_i} z(\mathbf{w}, \mathbf{c}_i)}{(\sum_{\mathbf{c}'_i \in \mathbf{C}_i} z(\mathbf{w}, \mathbf{c}'_i)) (\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in \mathbf{C}_k} z(\mathbf{w}, \mathbf{c}'_k))} \\
&= \sum_{\substack{k=1 \\ k \neq i}}^{N_{\text{classes}}} \sum_{\mathbf{c}_k \in \mathbf{C}_k} \frac{z(\mathbf{w}, \mathbf{c}_k)}{(\sum_{\mathbf{c}'_i \in \mathbf{C}_i} z(\mathbf{w}, \mathbf{c}'_i)) (\sum_{k=1}^{N_{\text{classes}}} \sum_{\mathbf{c}'_k \in \mathbf{C}_k} z(\mathbf{w}, \mathbf{c}'_k))}.
\end{aligned}$$

Now, summing over all samples $\mathbf{x} \in Y$, normalizing with $|Y|$ and taking the additive inverse yields the desired result.

When using mixup, the right hand side of the last equation needs to be replaced with a weighted sum of two terms, each corresponding to one of the two classes that are mixed-up, because there are $i_1, i_2 \in \{1, \dots, N_{\text{classes}}\}$ such that $\text{lab}(\mathbf{x})_{i_1} \neq 0 \neq \text{lab}(\mathbf{x})_{i_2}$. Otherwise, the proof is exactly the same. In conclusion, the proven result still holds for mixed-up samples but includes two similar terms instead of one term. \square

Note that this theorem explicitly shows that the intra-class compactness losses belonging to different sub-clusters of the same class are weighted with softmax probabilities that indicate to which sub-cluster an embedding belongs to. This allows that embeddings belong to a single sub-cluster by setting the probability belonging to one sub-cluster close to one and all other probabilities close to zero. Without these probabilistic weights, each embedding would be pulled towards the mean of all sub-clusters of the corresponding class to minimize the mean distance, which essentially means that only a single sub-cluster, i.e. one center, would be used for each class.

Using Theorem 3.5, a short proof for Corollary 3.3 will now be provided.

Proof of Corollary 3.3. The proof of Theorem 3.5 does not depend on the exact structure of the dynamically adaptive scale parameter and thus also holds for the standard AdaCos loss by replacing \hat{s} with \tilde{s} and using only a single sub-cluster for each class. \square

3.4.3 Comparison of backends

According to Lemma 3.1, the cosine distance is equivalent to using a Gaussian with a spherical covariance matrix. Hence, using a GMM with a full covariance matrix and possibly multiple components is a generalization of using the cosine distance and may perform better in case the distribution of the normal samples is not spherical. This is illustrated in Figure 10. Note that this illustration is exaggerated because, by definition, an angular margin loss tries to ensure that the distribution for each class is roughly spherical. However, the resulting distributions may still be more complex, especially in higher-dimensional spaces.

Next, the choice of a suitable backend shall be investigated experimentally. For all experiments with Gaussians or GMMs, the implementation provided by scikit-learn [176] is used.

The performances obtained with different backends can be found in Table 6. As expected, using the softmax output of the embedding model instead of the cosine distance performs worst. The reason is that a softmax function models a posterior distribution over the normal classes and thus is not aiming at detecting out-of-distribution samples, i.e. anomalies. PLDA, as implemented in [208], performs slightly better but still worse than all the other backends. Since they are strongly related to each other or even equivalent, all backends using cosine distance or Gaussians with a spherical or diagonal covariance matrix perform very similarly. When using full covariance matrices, an improvement in performance can be observed and thus Gaussians or GMMs with full covariance matrices are used as a backend in the following experiments.

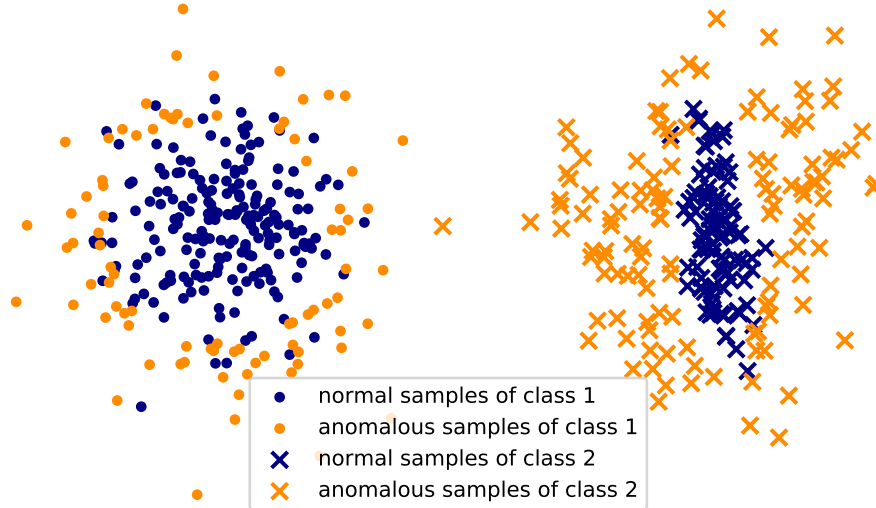


Figure 10: Scatter plot of normal and anomalous data belonging to two different classes. Both classes can be easily separated by measuring the distance to the respective class means. For class 1, anomalies can also be detected reasonably well. However, for class 2 only measuring the distance to the mean does not work well because the data is not distributed spherically. © 2021 IEEE

Table 6: Arithmetic means of **AUC-ROCs** and **pAUCs** for different machine types obtained with different backends. © 2021 IEEE

backend	development set		evaluation set	
	AUC-ROC	pAUC	AUC-ROC	pAUC
softmax output of embedding model	87.20%	81.70%	89.55%	83.79%
log-likelihood ratio of two-covariance PLDA	88.25%	82.18%	90.90%	84.32%
cosine distance to mean	88.71%	82.12%	91.13%	84.40%
mean of cosine distances to 10 closest samples	88.69%	82.12%	91.10%	84.38%
Gaussian (spherical covariance)	88.69%	82.12%	91.11%	84.38%
Gaussian (diagonal covariance)	88.71%	82.16%	91.12%	84.39%
Gaussian (full covariance)	89.13%	82.59%	91.43%	84.47%

Table 7: Arithmetic means of **AUC-ROCs** and **pAUCs** obtained with mixup and the sub-cluster AdaCos loss using only a single center per class. © 2021 IEEE

mixup	loss	development set		evaluation set	
		AUC-ROC	pAUC	AUC-ROC	pAUC
	AdaCos	86.96%	80.68%	89.63%	82.47%
	sub-cluster AdaCos	diverges	diverges	diverges	diverges
X	AdaCos	89.13%	82.59%	91.43%	84.47%
X	sub-cluster AdaCos	91.60%	85.01%	91.64%	83.93%

3.4.4 Utilizing mixup

Mixup is known to be very effective for **CSC** tasks. However, it is not clear whether the same benefits carry over to **ASD**. In addition, the definition of the dynamically adaptive scale parameter of the sub-cluster AdaCos loss, which depends on using mixup, still needs to be justified. Therefore, the effect of using mixup for **ASD** with angular margin losses is investigated by comparing the performances obtained with and without mixup. The results can be found in **Table 7** and the following three observations can be made: First, using mixup improves the performance regardless of the angular margin loss being used. As applying mixup is simple yet effective, there appears to be no reason to not use it for **ASD**. Second, the dynamically adaptive scale parameter of sub-cluster AdaCos loss performs better than the original one of the AdaCos loss, which justifies the particular definition of the scale parameter. Third, when not using mixup, the sub-cluster AdaCos loss diverges. The reason is given in the following lemma.

Lemma 3.6. *When not using mixup, the dynamically adaptive scale parameter $\hat{s}^{(t)}$ of the sub-cluster AdaCos loss grows exponentially.*

Proof. After a few training iterations without mixup, i.e. for $t > t_0 \in \mathbb{N}$, most training samples will have a very small angle to the centers of their corresponding class and therefore $\sum_{c \in C_i} \text{sim}(\phi(\mathbf{x}, \mathbf{w}^{(t)}), \mathbf{c}) > 1$ for all $i \in \{1, \dots, N_{\text{classes}}\}$ and $\mathbf{x} \in Y^{(t)} \cap Y_i$. Furthermore, as empirically shown in [274], on average $\alpha(\phi(\mathbf{x}, \mathbf{w}^{(t)}), \mathbf{c}) < \frac{\pi}{2}$ and thus $\text{sim}(\phi(\mathbf{x}, \mathbf{w}^{(t)}), \mathbf{c}) > 0$ for most $\mathbf{x} \in Y$ and $\mathbf{c} \in \mathbb{R}^D$.

Hence, by using the fact that the logarithm is a concave function and applying Jensen’s inequality it holds that

$$\begin{aligned}
\hat{s}^{(t)} &= \frac{\text{sim}_{\max}^{(t)} + \log \hat{\mathbf{B}}_{\text{avg}}^{(t)}}{\cos\left(\min\left(\frac{\pi}{4}, \hat{\alpha}_{\text{med}}^{(t)}\right)\right)} \geq \text{sim}_{\max}^{(t)} + \log \hat{\mathbf{B}}_{\text{avg}}^{(t)} \\
&= \log\left(\frac{1}{N_{\text{batch}}} \sum_{\mathbf{x} \in Y^{(t)}} \sum_{j=1}^{N_{\text{classes}}} \sum_{\mathbf{c} \in C_j} \exp\left(\hat{s}^{(t-1)} \text{sim}(\phi(\mathbf{x}, \mathbf{w}^{(t)}), \mathbf{c})\right)\right) \\
&\geq \frac{1}{N_{\text{batch}}} \sum_{\mathbf{x} \in Y^{(t)}} \sum_{j=1}^{N_{\text{classes}}} \sum_{\mathbf{c} \in C_j} \hat{s}^{(t-1)} \text{sim}(\phi(\mathbf{x}, \mathbf{w}^{(t)}), \mathbf{c}) \\
&> \hat{s}^{(t-1)} \underbrace{\left(1 + \frac{1}{N_{\text{batch}}} \sum_{\mathbf{x} \in Y^{(t)}} \sum_{\substack{j=1 \\ \text{lab}(\mathbf{x})_j \neq 1}}^{N_{\text{classes}}} \sum_{\mathbf{c} \in C_j} \text{sim}(\phi(\mathbf{x}, \mathbf{w}^{(t)}), \mathbf{c})\right)}_{>0}
\end{aligned}$$

showing that $\hat{s}^{(t)}$ grows exponentially when not using mixup. Note that this inequality does not hold if only mixed-up samples are used for training. The reason is that the mixed-up samples do not have a very high cosine similarity with all of their corresponding class centers because they are positioned between classes and AdaCos increases the margin between classes. \square

3.4.5 Determining the number of sub-clusters

In [Table 8](#), the effect of using a different number of sub-clusters is investigated experimentally. It can be seen that using multiple sub-clusters significantly improves the performance, especially on the evaluation set. This justifies the definition of the sub-cluster AdaCos loss. Overall, the best results are obtained with 32 sub-clusters and using more sub-clusters degrades performance. Hence, the optimal number of sub-clusters is neither too high nor too low and thus is an additional hyperparameter to be tuned. Another observation is that using a [GMM](#) with components equal to the number of sub-clusters leads to slightly better performance than using a single Gaussian because the underlying distribution can be better represented. Note that using sub-clusters also allows the model to handle potential outliers or noisy samples contained in the training set by assigning them to small sub-clusters as done with the sub-center ArcFace loss [\[37\]](#).

3.4.6 Replacing embeddings with input data statistics

An alternative to training an embedding model with the goal of obtaining simple representations of the data is to compute statistics over the input features such as the temporal mean or temporal maximum. These statistics have the advantage that they do not require any training, yet, they may still contain useful information

Table 8: Arithmetic means of **AUC-ROCs** and **pAUCs** obtained with the sub-cluster AdaCos loss for different numbers of centers per class. © 2021 IEEE

number of sub-clusters	backend	development set		evaluation set	
		AUC-ROC	pAUC	AUC-ROC	pAUC
1	Gaussian	91.60%	85.01%	91.64%	83.93%
2	Gaussian	90.97%	82.54%	92.08%	85.08%
4	Gaussian	91.54%	83.53%	92.62%	84.31%
8	Gaussian	91.61%	85.24%	92.99%	85.74%
16	Gaussian	91.85%	85.61%	93.98%	88.27%
32	Gaussian	92.22%	85.69%	94.56%	87.51%
64	Gaussian	91.39%	83.58%	93.85%	85.43%
1	GMM	91.60%	85.01%	91.64%	83.93%
2	GMM	91.07%	82.70%	92.20%	85.60%
4	GMM	91.67%	83.70%	92.64%	84.35%
8	GMM	91.85%	85.46%	93.13%	86.07%
16	GMM	92.10%	85.84%	94.08%	88.59%
32	GMM	92.57%	86.37%	94.69%	87.90%
64	GMM	92.03%	84.06%	94.16%	86.19%

for detecting anomalous samples. The main difference to trained embeddings is that the information contained in these representations is not necessarily useful to discriminate between the classes of the auxiliary task. This may have a positive or negative impact on detecting anomalous samples and will now be investigated experimentally. To this end, statistics of all features belonging to individual data samples are calculated and treated as if they were trained embeddings. To compute an anomaly score, a **GMM** with a single Gaussian component is used as using more components did not improve the performance.

The experimental results can be found in [Table 9](#). It can be seen that the overall performance of the statistical representations is much worse than the one obtained with trained embeddings. This is also to be expected as using an auxiliary classification task allows the embeddings to ignore the background noise whereas statistical representations are more strongly affected by noise. However, these statistical representations lead to surprisingly good performance. The temporal maximum works well for the machine type “valve” and the temporal mean works well for the two machine types “ToyCar” and “ToyConveyor”. For “ToyConveyor”, the performance is even much better than the learned embeddings. A likely reason is that the auxiliary classification task is too simple meaning that the machines of this machine type are easily recognized and thus the embeddings do not capture enough infor-

Table 9: Mean **AUC-ROCs** and **pAUCs** per machine type obtained with different representations. When using the combined representations, only the learned representations are used, except for machine type ToyConveyor where the temporal mean is used instead. © 2021 IEEE

representation	machine type	development set		evaluation set	
		AUC-ROC	pAUC	AUC-ROC	pAUC
temporal mean	fan	80.73%	66.16%	95.32%	80.62%
temporal maximum	fan	64.59%	51.48%	78.98%	57.70%
learned	fan	87.61%	77.93%	97.60%	93.24%
temporal mean	pump	82.99%	68.50%	88.24%	70.36%
temporal maximum	pump	70.13%	59.17%	68.96%	55.08%
learned	pump	94.71%	88.91%	96.76%	88.30%
temporal mean	slider	87.46%	63.95%	72.16%	53.08%
temporal maximum	slider	93.69%	76.69%	90.55%	70.65%
learned	slider	99.55%	97.63%	97.61%	89.46%
temporal mean	valve	55.59%	50.23%	54.86%	52.09%
temporal maximum	valve	98.54%	93.08%	96.35%	88.18%
learned	valve	98.63%	94.62%	98.81%	95.80%
temporal mean	ToyCar	94.10%	80.94%	91.54%	76.87%
temporal maximum	ToyCar	68.36%	53.85%	70.31%	54.90%
learned	ToyCar	96.37%	91.64%	95.99%	91.93%
temporal mean	ToyConveyor	85.78%	67.76%	91.74%	78.13%
temporal maximum	ToyConveyor	57.51%	50.39%	65.40%	53.06%
learned	ToyConveyor	73.89%	61.22%	81.37%	68.64%
temporal mean	all	80.91%	66.19%	82.31%	68.52%
temporal maximum	all	76.25%	64.71%	78.42%	63.26%
learned	all	92.57%	86.37%	94.69%	87.90%
combined	all	94.21%	87.13%	96.42%	89.24%

mation to discriminate between normal and anomalous samples. For the machine types for which one statistic works well, the other statistic performs very poorly. Hence, while these statistical representations may lead to a good performance, they need to be carefully chosen for each machine type. To obtain the best possible performance, a combined model is designed by using the learned embeddings for each machine type except “ToyConveyor” for which the temporal mean is used

instead. This approach of using statistics of input features as representations was later extended in [70] by using global weighted ranking pooling [109], i.e. applying a geometric sequence of weights to time frames of a log-Mel spectrogram sorted according to their energy over all frequency bins. Global weighted ranking pooling is a generalization of both statistics used above and led to a good overall performance without training any model except for choosing the weights. However, note that determining these weights for individual machine types requires access to anomalous data for each machine type and thus is highly impractical for any given application and in fact not possible in a truly semi-supervised setting. This is the reason why no additional experiments with this approach are carried out in this thesis.

3.4.7 Comparison to other published systems

To show that the presented ASD system performs well, its performance is compared to the five top-performing systems of the DCASE2020 Challenge in Figure 11. It can be seen that the system outperforms all other systems and thus reaches a new state-of-the-art performance. Since all systems except the one submitted by Primus [183] are ensembles, an ensemble was also created here to allow for a fair comparison. The ensemble is obtained by training seven variations of the single model, each with a different number of sub-clusters ranging from 2^0 to 2^6 , and using the sum of the anomaly scores obtained with each system as the ensembled anomaly score. As a result, the ensembled system reaches an AUC-ROC of 97% and a pAUC of 91.24% and thus performs even better than the proposed single model system.

3.5 SUMMARY

In this chapter, different loss functions for training an embedding model were compared theoretically and empirically. More concretely, the compactness loss and the AdaCos loss as representatives of one-class and angular margin losses were compared. It was shown that both are very closely related as minimizing an angular margin loss minimizes the intra-class compactness losses for each class while also maximizing all inter-class compactness losses. Additionally, it was shown experimentally that the ASD performance is much better when using a classification task in a joint embedding space instead of not using meta information as classes or individual embedding spaces learned with one-class losses. This helps to closely monitor the target sounds to detect anomalous signal components and largely ignore the background noise.

The other main content of this chapter has been the presentation of the sub-cluster AdaCos loss, which has a similar relation to the compactness loss as the AdaCos loss. The sub-cluster AdaCos loss is an extension of AdaCos that uses multiple centers for each class to enable the embedding model to utilize sub-classes

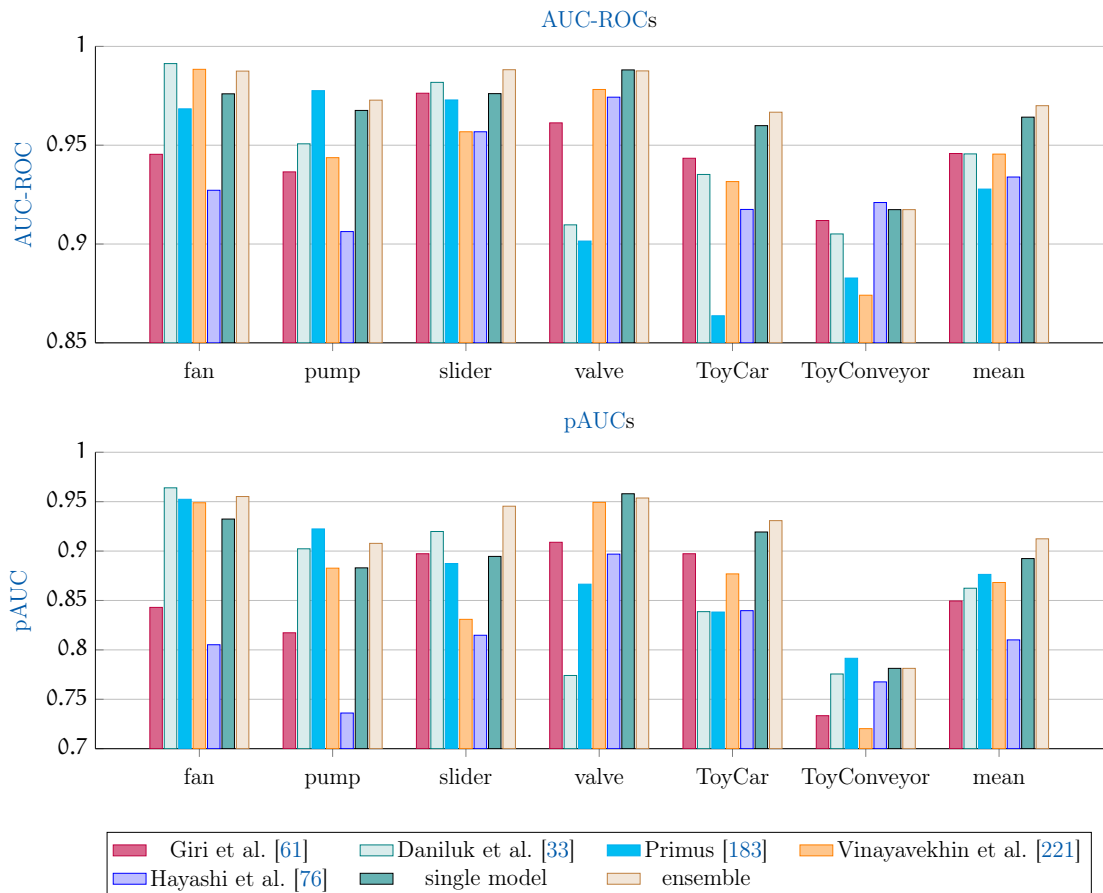


Figure 11: Comparison of the **AUC-ROCs** and **pAUCs** obtained on the evaluation set with the top five highest-ranked systems submitted to the DCASE2020 Challenge task 2, the proposed approach and an ensemble. The ensemble consists of the sum of all log-probabilities given by **GMMs** belonging to trained models of the proposed approach with a different number of sub-clusters, ranging from 2^0 to 2^6 . © 2021 IEEE

and learn more complex distributions for the normal data than simple Gaussian distributions. Using the sub-cluster AdaCos loss, mixup and a **GMM** with a full covariance matrix were all shown to significantly improve the **ASD** performance. As a result, the proposed **ASD** system and an ensemble reached a new state-of-the-art performance on the DCASE2020 dataset. In a last experiment, it was shown that using statistics of the input data, as for example the temporal mean of time-frequency representations, instead of learning embeddings yields surprisingly good results. Still, the performance is worse than when using the learned embeddings.

In total, the presented **ASD** system reached an **AUC-ROC** of 97% on the DCASE2020 dataset, which is close to optimal performance. Hence, the difficulty of the **ASD** task needs to be increased in order to obtain additional findings. This will be done in **Chapter 5** by utilizing datasets for acoustic machine condition mon-

itoring recorded in domain-shifted conditions. But before increasing the difficulty of the task, it will be investigated how to estimate good decision thresholds.

When using an ASD system in practice, decision thresholds need to be applied to the anomaly scores to distinguish between normal and anomalous samples. As stated in Section 2.11, several methods are available for estimating decision thresholds in a semi-supervised setting. The reader shall be reminded that only normal training data can be used for estimating the decision thresholds as already stated in Section 2.11. The reason is that anomalous training samples are not available and that test samples should be treated independently, i.e. one cannot utilize all (unlabeled) normal and anomalous test samples for estimating a threshold based on their assumed binary class distribution. Because of this, all estimation methods are based on the assumption that a threshold, which separates the extreme values of the anomaly scores belonging to normal training samples from the moderate values, is also a good choice for separating anomalous and normal samples. The goal of this chapter is to investigate how to robustly estimate good decision thresholds and how to take the difficulty of estimating a threshold into account when evaluating the performance of an ASD system.

This chapter is structured as follows: First, multiple methods for estimating a decision threshold will be compared experimentally. In a second section, the threshold-independent evaluation metric F_1 -EV, which in contrast to AUC-ROC also measures the difficulty of estimating a good decision threshold, will be presented and compared to other evaluation metrics.

4.1 CONTRIBUTIONS OF THE AUTHOR

The sections of this chapter are largely based on the following key publications:

- Kevin Wilkinghoff and Alessia Cornaggia-Urrigshardt. “On choosing decision thresholds for anomalous sound detection in machine condition monitoring.” In: *24th International Congress on Acoustics*. The Acoustical Society of Korea, 2022.
- Kevin Wilkinghoff and Keisuke Imoto. “F1-EV Score: Measuring the Likelihood of Estimating a Good Decision Threshold for Semi-Supervised Anomaly Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 256–260. DOI: [10.1109/ICASSP48485.2024.10446011](https://doi.org/10.1109/ICASSP48485.2024.10446011).

For publications that are not single-authored, individual contributions of the thesis author and all co-authors to these publications are stated in Section A.1. If not stated otherwise, the content listed in the following paragraph is the sole contribution of the thesis author.

Section 4.2 is based on [252]. Alessia Cornaggia-Urrigshardt helped to implement the different approaches for estimating decision thresholds. Section 4.3 is based on [260]. Keisuke Imoto proposed to include Section 4.3.4 including Figure 20. Furthermore, he provided the anomaly scores and decision thresholds of the systems submitted to the DCASE2023 ASD Challenge, which are used as a dataset as described in Section 4.3.2.

4.2 ESTIMATING A DECISION THRESHOLD

In Section 2.11, multiple methods for estimating a decision threshold were presented. However, when designing an ASD system two questions remain: “Which of these methods should be used?” and “To which anomaly scores should such a method be applied?”. The goal of this section is to answer both questions. For all estimation methods evaluated in this section, 90th percentiles and the hyperparameter settings $\beta_{SD} = 1.28$, $\beta_{MAD} = 2$, $\beta_{IQR} = 0.5$, $\nu_{SVM} = 0.1$ as well as two iterations for MST were used as similar settings are also used in the literature and these particular settings led to reasonable results.

4.2.1 Performance comparison of different estimation methods

To compare the performance of the estimation methods, the F_1 score resulting from applying an estimation method to the anomaly scores obtained with the same ASD system as presented in Section 3.2.2 is used. To reduce the variance of the results, each experiment is repeated five times by retraining the ASD system and estimating a decision threshold with each method. In addition to the estimation methods, optimal decision thresholds are determined by manually varying them to optimize the performance on the test splits of the development and evaluation set. These optimal thresholds are used to analyze how close the performance obtained with an estimated decision threshold is to the best possible performance.

The F_1 scores obtained on the test split of the development set and of the evaluation set can be found in Table 10 and Table 11, respectively.

Depending on the machine type, one can observe that the absolute performances are strongly varying. This is also true for the optimum performance and therefore drawing conclusions by analyzing the absolute F_1 scores is difficult. A better approach is to measure how close the performance of an estimated decision threshold is to the performance obtained with an optimal decision threshold. This was done in a second experiment by calculating the ratio between both performances. The results are depicted in Figure 12. It can be observed that most estimation methods perform equally well except for GESD, which led to worse performance on the development and evaluation set, and GDP as well as MST-GDP, which led to slightly worse performance on the development set. Furthermore, the iterative approaches (GESD, cleverSD and MST) perform slightly better than their non-

Table 10: Comparison of F_1 scores obtained with different threshold estimation methods on the *development set* of the DCASE2020 dataset. Means of F_1 scores taken over all machine IDs belonging to single machine types obtained with five independent trials are shown. Highest F_1 score among different methods for each machine type is highlighted in bold.

method	machine type						mean
	fan	pump	slider	ToyCar	ToyConveyor	valve	
GDP	0.832	0.823	0.911	0.808	0.618	0.879	0.812
HP	0.753	0.837	0.951	0.848	0.673	0.898	0.827
SD	0.751	0.839	0.951	0.850	0.672	0.900	0.827
MAD	0.762	0.844	0.951	0.854	0.670	0.902	0.830
IQR	0.790	0.843	0.939	0.837	0.677	0.897	0.831
OCSVM	0.753	0.837	0.950	0.849	0.673	0.898	0.827
GESD	0.662	0.816	0.968	0.864	0.594	0.870	0.796
cleverSD	0.836	0.835	0.919	0.821	0.663	0.885	0.826
MST-GDP	0.864	0.810	0.888	0.773	0.611	0.865	0.802
MST-HP	0.816	0.836	0.926	0.814	0.663	0.888	0.824
MST-SD	0.827	0.833	0.919	0.806	0.659	0.882	0.821
MST-MAD	0.797	0.849	0.940	0.847	0.676	0.898	0.835
MST-IQR	0.833	0.834	0.916	0.806	0.659	0.883	0.822
MST-OCSVM	0.816	0.836	0.927	0.815	0.663	0.887	0.824
optimum	0.926	0.889	0.985	0.892	0.689	0.919	0.883

Table 11: Comparison of F_1 scores obtained with different threshold estimation methods on the *evaluation set* of the DCASE2020 dataset. Means of F_1 scores taken over all machine IDs belonging to single machine types obtained with five independent trials are shown. Highest F_1 score among different methods for each machine type is highlighted in bold.

method	machine type						mean
	fan	pump	slider	ToyCar	ToyConveyor	valve	
GDP	0.900	0.861	0.889	0.601	0.635	0.671	0.759
HP	0.938	0.892	0.923	0.584	0.597	0.656	0.765
SD	0.937	0.893	0.921	0.577	0.600	0.656	0.764
MAD	0.939	0.890	0.919	0.572	0.598	0.651	0.761
IQR	0.941	0.894	0.920	0.600	0.619	0.665	0.773
OCSVM	0.938	0.892	0.923	0.583	0.596	0.656	0.765
GESD	0.910	0.882	0.900	0.513	0.496	0.616	0.719
cleverSD	0.941	0.874	0.909	0.618	0.643	0.679	0.777
MST-GDP	0.907	0.852	0.885	0.638	0.663	0.692	0.773
MST-HP	0.943	0.885	0.915	0.618	0.636	0.676	0.779
MST-SD	0.943	0.879	0.911	0.624	0.642	0.683	0.780
MST-MAD	0.941	0.892	0.919	0.592	0.619	0.664	0.771
MST-IQR	0.943	0.875	0.910	0.626	0.643	0.684	0.780
MST-OCSVM	0.943	0.885	0.915	0.618	0.636	0.676	0.779
optimum	0.965	0.944	0.959	0.759	0.725	0.784	0.856

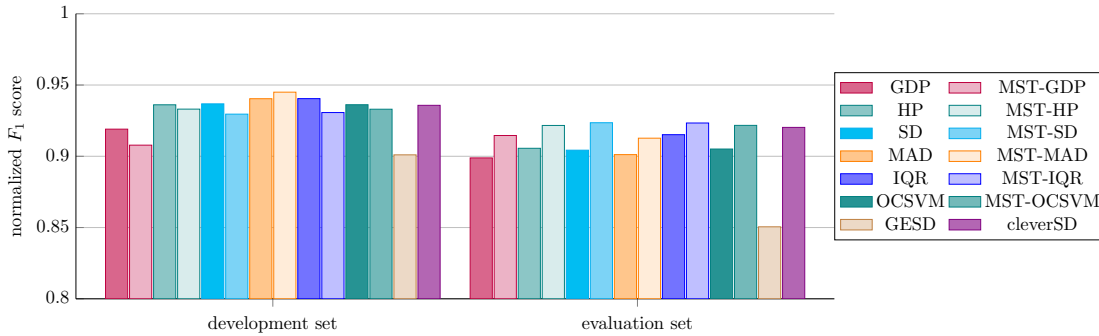


Figure 12: Normalized F_1 scores obtained with different threshold estimation methods on the DCASE2020 dataset. Means of normalized F_1 scores taken over all machine IDs belonging to single machine types obtained with five independent trials are shown.

iterative counterparts and thus should be the preferred choice when estimating decision thresholds.

4.2.2 Choosing a set of observed anomaly scores

Only the normal training samples are available to calculate anomaly scores and thus the same scores that have been used to train the ASD system need to be used to estimate a decision threshold. However, an alternative is to first divide the normal training samples into two disjoint sets and use one set to train the ASD system and the other set to estimate a decision threshold. The idea is to use previously unseen samples when estimating the decision thresholds as this may generate less optimistic and thus more realistic anomaly scores, which are expected to be more similar to the anomaly scores of the test samples than the ones used for training the model.

When using only a subset of the normal training samples to train the ASD system, it is expected that the performance will degrade because less information is provided. As a first experiment, it will be investigated how much normal data should at least be used to train the system and how much data can be used for estimating thresholds. This is done by computing the optimal F_1 score for models that have only been trained with a subset of the normal training samples. The results, depicted in Figure 13, show that the degradation of performance is far less severe as one would expect and is only noticeable when using less than 60% of the data for training. Moreover, even when using only 5% of training samples the degradation in performance is relatively small. A possible explanation is that the variation of fully-functioning machine sounds is not very strong when ignoring the background noise and not changing any parameter settings and thus their normal behaviour can be captured with relatively few data samples.

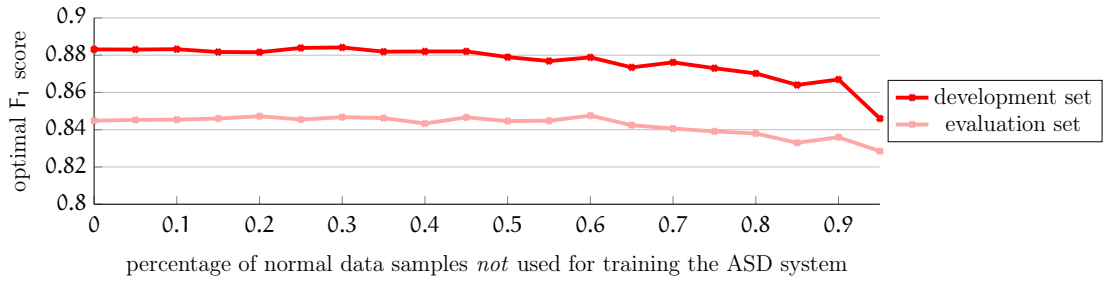


Figure 13: Optimal F_1 scores obtained on the DCASE2020 dataset when using a varying percentage of normal data samples not used for training the ASD system. Means of optimal F_1 scores taken over all machine IDs belonging to single machine types obtained with five independent trials are shown.

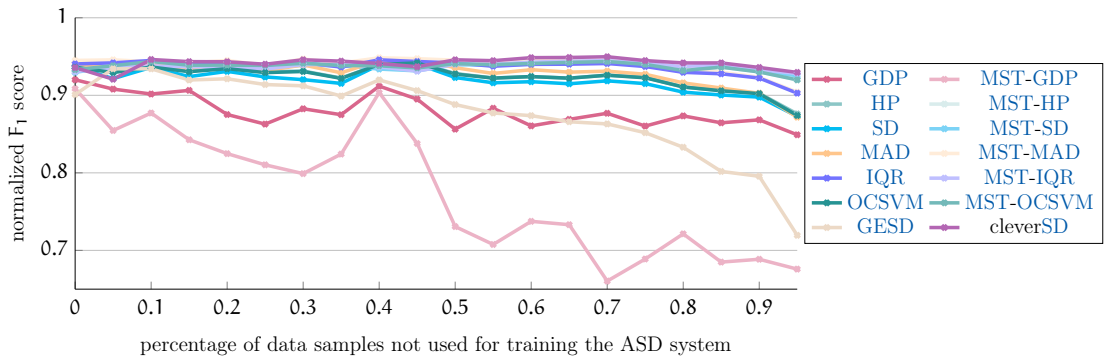


Figure 14: Normalized F_1 scores obtained with different threshold estimation methods on the development set of the DCASE2020 dataset when using a varying percentage of normal data samples not used for training the ASD system. Means of normalized F_1 scores taken over all machine IDs belonging to single machine types obtained with five independent trials are shown.

In a second experiment, the normalized F_1 scores were calculated for different partitions of the normal training samples into a set used for training the system and a set for estimating the thresholds. The results are depicted in Figure 14 and Figure 15.

The following observations can be made: First, most methods have a relatively stable performance except for GDP, MST-GDP and GESD. Second, iterative approaches perform slightly better than non-iterative estimation methods. All of these observations are consistent with the findings of the previous subsection. Thirdly and most importantly, there is no clearly visible improvement in performance when using only a subset of the normal samples for training the model. Actually, the absolute performance is decreasing because the optimal F_1 scores are also decreasing (cf. Figure 13). As a last observation, the difference in performance between the estimation methods increases the less data is used for training the system. In conclusion, only using a part of the normal samples for training and

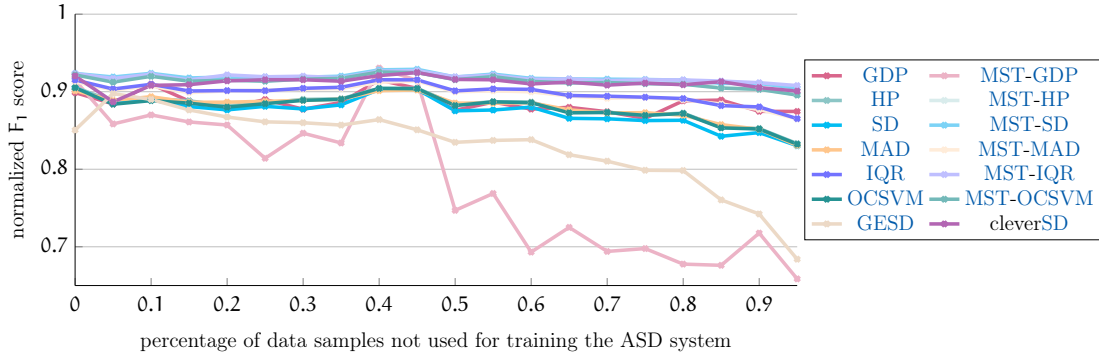


Figure 15: Normalized F_1 scores obtained with different threshold estimation methods on the evaluation set of the DCASE2020 dataset when using a varying percentage of normal data samples not used for training the ASD system. Means of normalized F_1 scores taken over all machine IDs belonging to single machine types obtained with five independent trials are shown.

the other part for estimating the decision threshold has no benefits and one can simply use all data for training the model.

4.3 F_1 -EV SCORE

An AUC - ROC score equal to one means that an optimal decision threshold perfectly separating the anomaly scores of all normal samples from the ones of the anomalous samples exists (see Theorem 2.10). Despite having such a maximal AUC - ROC score, the cardinality of the set of optimal decision thresholds and thus also the difficulty of estimating this optimal threshold may strongly vary for different sets of anomaly scores. This is illustrated with toy examples in Figure 16. In practice, estimating a good decision threshold is important to be able to separate normal and anomalous samples. Therefore, the difficulty of estimating such a threshold should be included to fully measure the performance of an ASD system. A naïve solution to this problem is to use an additional threshold-dependent performance measure such as the F_1 score. However, threshold-dependent metrics depend on the specific choice of a threshold and thus are subjective. Furthermore, an objective comparison of ASD systems requires a well-defined ranking of performances, which is difficult to define when using multiple performance metrics at once. This motivates the definition of the F_1 -EV score, which is a threshold-independent performance measure that incorporates the difficulty of estimating a good decision threshold.

4.3.1 Definition

The idea of the F_1 -EV score is to calculate the expected value (EV) of a random variable that models the F_1 score of an ASD system for varying decision thresholds.

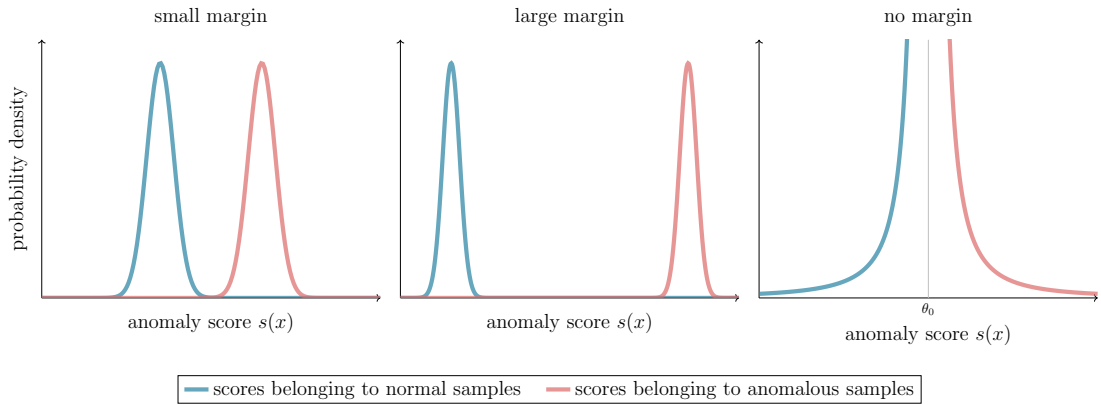


Figure 16: Toy examples of perfectly separable anomaly score distributions, each with an AUC - ROC equal to 1. On the left, the margin between normal and anomalous scores is small. In the center, the margin between the distributions is large and thus estimating a good decision threshold is much easier. On the right, the only optimal decision threshold is $\theta = \theta_0$, which is a null set with measure zero, and thus estimating this threshold also has a likelihood of zero when assuming a continuous distribution with uncountable support for estimated thresholds. Note that when estimating a decision threshold, only a finite number of anomaly scores belonging to the distribution of normal samples are available and usually both distributions are more complex and overlap. This is the reason why estimating a good decision threshold is highly non-trivial. © 2024 IEEE

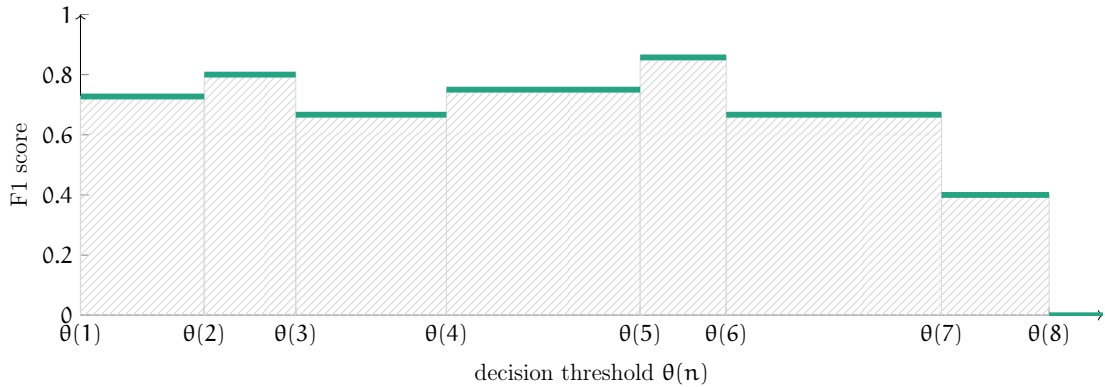


Figure 17: Example of computing F_1 -EV. For illustration purposes only very few decision thresholds are shown. © 2024 IEEE

This is done by calculating the F_1 score for all possible thresholds and thus choosing a specific decision threshold is not required, i.e. the F_1 -EV score is threshold-independent. A finite Riemann sum is used to calculate the EV given by the area under the F_1 score function as illustrated in Figure 17. Note that according to [35], computing the area under the precision-recall curve by using linear interpolations such as the trapezoidal rule leads to over-optimistic approximations. The F_1 score

is the harmonic mean of precision and recall, which is the reason why a Riemann sum is used instead of the trapezoidal rule. Since the F_1 score function is piecewise constant with respect to the decision threshold, using more points than the anomaly scores of the provided samples when computing the area does not improve the accuracy of the approximation. The F_1 -EV score will now be formally defined.

Definition 4.1 (F_1 -EV score). Let $(\theta_{\text{sorted}}(\mathbf{k}))_{\mathbf{k}=1, \dots, |\mathcal{X}_{\text{test}}|} \subset \mathbb{R}$ denote a monotonically increasing sequence of decision thresholds, which correspond to the sorted anomaly scores obtained with the test samples. Define the normalized distance between two decision thresholds as

$$\bar{\Delta}_{\text{diff}}\theta_{\text{sorted}}(\mathbf{k}) := \frac{\theta_{\text{sorted}}(\mathbf{k} + 1) - \theta_{\text{sorted}}(\mathbf{k})}{\theta_{\text{sorted}}(|\mathcal{X}_{\text{test}}|) - \theta_{\text{sorted}}(1)}. \quad (48)$$

Then, the F_1 -EV score is defined as

$$F_1\text{-EV}(\mathcal{X}_{\text{test}}, \text{score}, \theta_{\text{sorted}}) := \sum_{\mathbf{k}=1}^{|\mathcal{X}_{\text{test}}|-1} F_1(\mathcal{X}_{\text{test}}, \text{score}, \theta_{\text{sorted}}(\mathbf{k})) \bar{\Delta}_{\text{diff}}\theta_{\text{sorted}}(\mathbf{k}) \in [0, 1] \quad (49)$$

with higher values indicating better performance.

Remark. When assuming a uniform distribution for randomly choosing decision thresholds in the interval $[\theta_{\text{sorted}}(1), \theta_{\text{sorted}}(|\mathcal{X}_{\text{test}}|)]$, $\bar{\Delta}_{\text{diff}}\theta_{\text{sorted}}(\mathbf{k})$ corresponds to the likelihood of choosing a decision threshold yielding the same F_1 score as $\theta_{\text{sorted}}(\mathbf{k})$. Hence, the F_1 -EV curve is indeed the expected value of a random variable modeling the F_1 score of an ASD system.

To calculate the F_1 -EV score, one has to provide a range $[\theta_{\min}, \theta_{\max}] \subset \mathbb{R}$ to which all possible estimated decision thresholds belong to. Otherwise, it would always be equal to zero for a finite set of anomaly scores. In the previous definition, this range was defined by using the values $\theta_{\text{sorted}}(1)$ and $\theta_{\text{sorted}}(|\mathcal{X}_{\text{test}}|)$ as lower and upper bounds, respectively. However, many values contained in this interval are unlikely to be chosen as decision thresholds. This is particularly true if the anomaly scores belonging to the test set contain a few outliers that strongly skew the results. Hence, robustly chosen boundaries may be beneficial to improve the F_1 -EV score as a performance metric. To this end, the lower bound $\theta_{\min} \in \mathbb{R}$ is chosen to be close to the mean of the normal test samples, assuming that one would not estimate a decision threshold that is much smaller than this value. The upper bound $\theta_{\max} \in \mathbb{R}$ is more difficult to choose because some large anomaly scores may be outliers having very high values. Instead the upper bound is chosen to be close to the center $\theta_{\text{opt}} \in \mathbb{R}$ of the set of all empirically optimal decision thresholds, assuming that ideally one would not estimate a decision threshold much larger than this value. Note that these boundaries are based on the anomaly scores belonging to the normal test samples to be independent from a particular set of training samples. Although one may argue that this leads to over-optimistic

results, it needs to be assumed that the normal samples of the training and test set follow the same distribution to be able to estimate a suitable decision threshold anyway. Furthermore, as shown in [Section 4.2.2](#), holding back a few normal training samples to estimate better decision thresholds does not improve the performance indicating that this assumption is valid. Formally, the bounded F₁-EV score is defined as follows.

Definition 4.2 (Bounded F₁-EV score). Let $(\theta_{\text{sorted}}(\mathbf{k}))_{\mathbf{k}=1, \dots, |\mathcal{X}_{\text{test}}|} \subset \mathbb{R}$ denote a monotonically increasing sequence of decision thresholds, which correspond to the sorted anomaly scores obtained with the test samples. Let $\beta_{\text{F}_1\text{-EV}} \in \mathbb{R}_+$ and $\theta_{\text{opt}} \in \mathbb{R}$ denote the center of all empirically optimal decision thresholds. Define

$$\begin{aligned} \theta_{\min} &:= \text{mean}((\theta_{\text{sorted}}(\mathbf{k}))_{\mathbf{k}=1, \dots, |\mathcal{X}_{\text{test}}|}) - \beta_{\text{F}_1\text{-EV}} \cdot \text{std}((\theta_{\text{sorted}}(\mathbf{k}))_{\mathbf{k}=1, \dots, |\mathcal{X}_{\text{test}}|}) \\ \theta_{\max} &:= \theta_{\text{opt}} + \beta_{\text{F}_1\text{-EV}} \cdot \text{std}((\theta_{\text{sorted}}(\mathbf{k}))_{\mathbf{k}=1, \dots, |\mathcal{X}_{\text{test}}|}) \end{aligned} \quad (50)$$

and set $\mathbf{k}_{\min} := \arg \min_{\mathbf{k}=1, \dots, |\mathcal{X}_{\text{test}}|} \{\theta_{\text{sorted}}(\mathbf{k}) : \theta_{\text{sorted}}(\mathbf{k}) > \theta_{\min}\}$ and $\mathbf{k}_{\max} := \arg \max_{\mathbf{k}=1, \dots, |\mathcal{X}_{\text{test}}|} \{\theta_{\text{sorted}}(\mathbf{k}) : \theta_{\text{sorted}}(\mathbf{k}) < \theta_{\max}\}$. Furthermore, set $\mathbf{K}_{\text{valid}} := \mathbf{k}_{\max} - \mathbf{k}_{\min} + 3$ and

$$\theta_{\text{bounded}}(\mathbf{k}) := \begin{cases} \theta_{\min} & \text{if } \mathbf{k} = 1 \\ \theta_{\text{sorted}}(\mathbf{k} + \mathbf{k}_{\min} - 2) & \text{if } 2 \leq \mathbf{k} \leq \mathbf{K}_{\text{valid}} - 1 \\ \theta_{\max} & \text{if } \mathbf{k} = \mathbf{K}_{\text{valid}} \end{cases} \quad (51)$$

for $\mathbf{k} = 1, \dots, \mathbf{K}_{\text{valid}}$. Then, the bounded F₁-EV score is defined as

$$\begin{aligned} & \text{F}_1\text{-EV}_{\text{bounded}}(\mathcal{X}_{\text{test}}, \text{score}, \theta_{\text{bounded}}) \\ & := \sum_{\mathbf{k}=1}^{\mathbf{K}_{\text{valid}}-1} \text{F}_1(\mathcal{X}_{\text{test}}, \text{score}, \theta_{\text{bounded}}(\mathbf{k})) \bar{\Delta}_{\text{diff}} \theta_{\text{bounded}}(\mathbf{k}) \in [0, 1] \end{aligned} \quad (52)$$

with

$$\bar{\Delta}_{\text{diff}} \theta_{\text{bounded}}(\mathbf{k}) := \frac{\theta_{\text{bounded}}(\mathbf{k} + 1) - \theta_{\text{bounded}}(\mathbf{k})}{\theta_{\text{bounded}}(\mathbf{K}_{\text{valid}}) - \theta_{\text{bounded}}(1)}. \quad (53)$$

4.3.2 Experimental setup

To experimentally evaluate the F₁-EV score, the anomaly scores and decision thresholds of all systems submitted to the ASD task of the DCASE2023 Challenge [44] were used. A detailed description of the DCASE2023 dataset can be found in [Section 5.2](#). For the experiments conducted in this section, it is sufficient to know that the dataset contains recordings of 14 different machine types. Each machine type is recorded under two different acoustic conditions, called source and target domain. During testing, it is unknown to which of these two domains

a given sample belongs to. Evaluations are done independently for each machine type using a single decision threshold for both domains. To compare the performances of ASD systems submitted to this challenge, the harmonic mean of the AUC-ROCs and pAUCs belonging to all 14 machine types was used. Therefore, choosing a decision threshold is not necessary and was entirely optional, which is the reason why some participants did not provide a decision threshold. For the experimental evaluations done here, only the F_1 scores of all submitted systems belonging to individual machine types that are greater than zero were used to not use invalid submissions. Furthermore, $\beta_{F_1-EV} = 0.2$ is used for the bounded F_1 -EV score.

4.3.3 *Experimental comparison with existing evaluation metrics*

Figure 18 and Figure 19 depict comparisons of different evaluation metrics in terms of the Pearson correlation coefficient (PCC). The following observations can be made: First of all, AUC-ROC has a low to moderate correlation with the F_1 scores resulting from the estimated (PCC = 0.503) and the optimal decision thresholds (PCC = 0.497). This affirms the motivation for introducing F_1 -EV as a new performance measure. Second, the F_1 -EV score has a very low correlation with AUC-ROC (PCC = 0.199) and a low correlation with the F_1 scores (PCC = 0.380 and PCC = 0.306). Third, the bounded F_1 -EV score has a high correlation with AUC-ROC (PCC = 0.748) and both F_1 scores (PCC = 0.696 and PCC = 0.732). This shows that the bounded F_1 -EV score works as intended and properly defined bounds for the F_1 -EV score are needed to obtain useful results. Although most estimation methods lead to decision thresholds with similar performance as shown in Section 4.2, it shall be emphasized that for some submissions the estimated decision thresholds may be far from optimal. For these submissions, the correlation between the bounded F_1 -EV and the F_1 score obtained with the estimated decision thresholds can be increased when using better estimates.

When inspecting Figure 18, one can see clusters looking similar to horizontal lines in all sub-figures belonging to F_1 scores of approximately two-thirds. To give an example, in sub-figure (a) this horizontal line ranges from an AUC-ROC of 0.4 to an AUC-ROC of 0.9. These lines look suspiciously wrong but can in fact be explained by the following: Since the recordings for each machine type belong to two different domains, it is possible that the provided decision thresholds yield almost perfect results for one domain while performing very poorly for the other domain. This means that either precision or recall are almost equal to 1 for both domains. Then the other value is close to 1 for only one domain while being close to 0 for the other domain and thus is approximately equal to 0.5 for both domains when assuming that both domains have approximately the same number of test samples. Since the F_1 score is the harmonic mean of precision and recall, this yields an F_1 score equal to two-thirds.

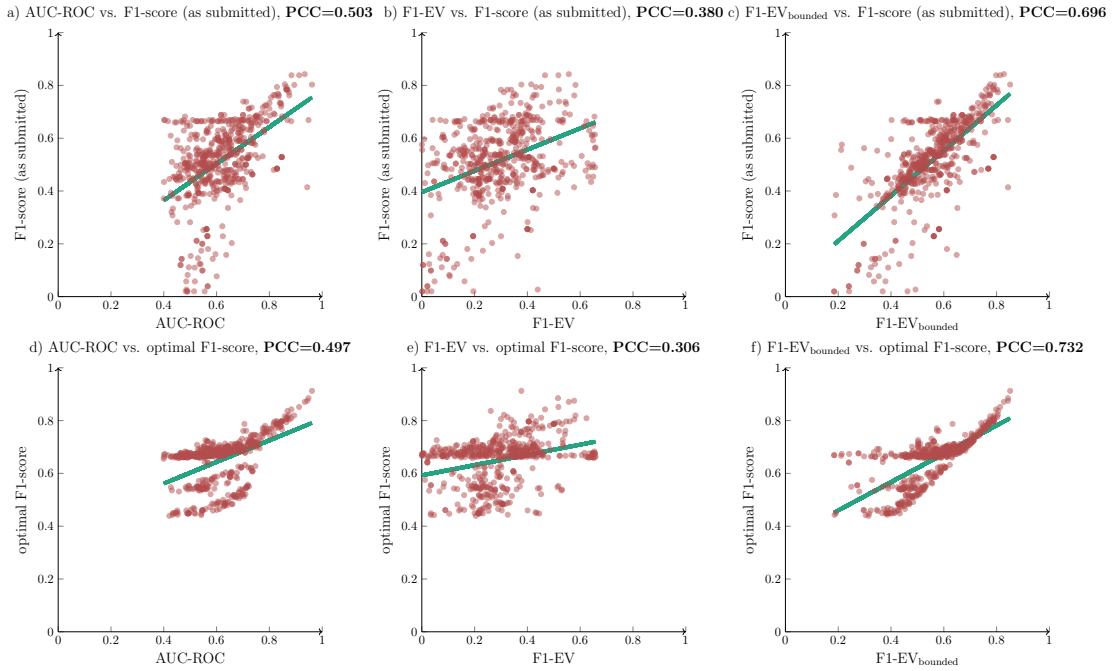


Figure 18: Comparison of several different performance measures computed on the evaluation set of task 2 of the DCASE2023 Challenge. In the top row, threshold-independent performance measures are compared to the F_1 score obtained with the submitted decision threshold. In the bottom row, threshold-independent performance measures are compared to the F_1 score obtained with an optimal decision threshold. © 2024 IEEE

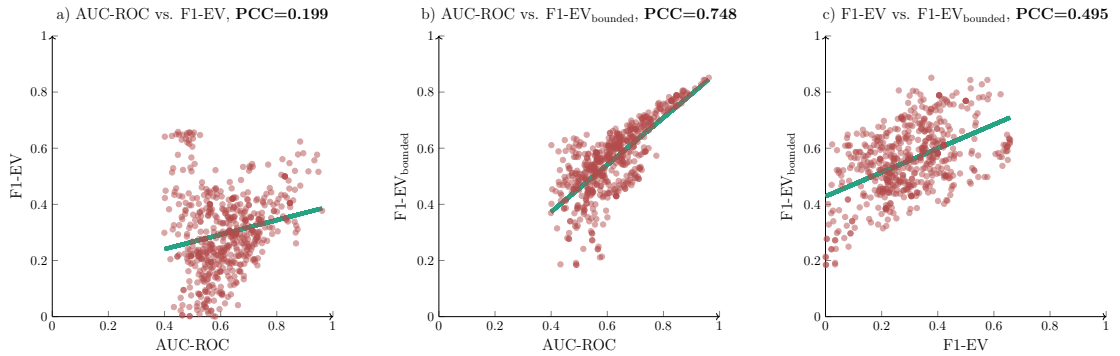


Figure 19: Comparison of threshold-independent performance measures computed on the evaluation set of task 2 of the DCASE2023 Challenge. © 2024 IEEE

4.3.4 Choosing the hyperparameter β_{F_1-EV}

In Figure 20, the PCCs between the bounded F_1 -EV score and other performance metrics are depicted for varying values of the hyperparameter β_{F_1-EV} . This allows to investigate the sensitivity with respect to β_{F_1-EV} and to provide recommendations for setting this hyperparameter. It can be seen that for $\beta_{F_1-EV} > 0.2$ the

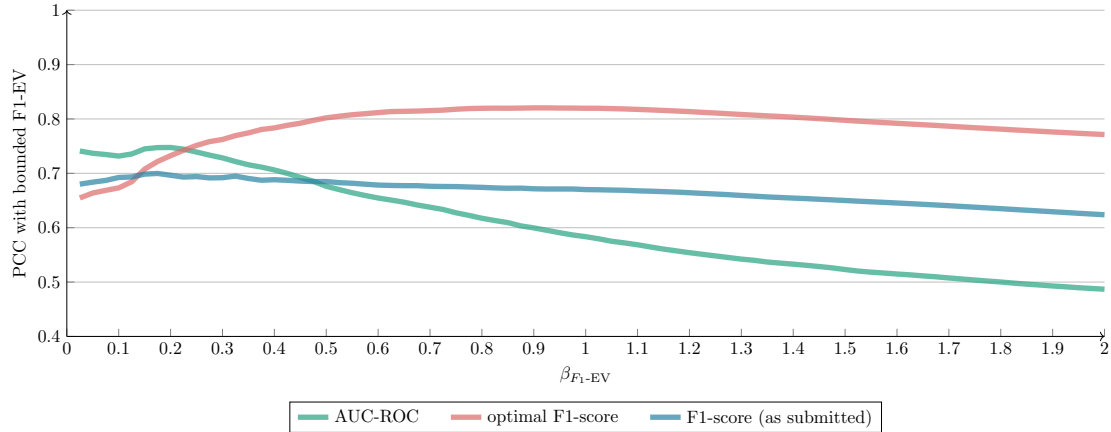


Figure 20: Sensitivity of the bounded F_1 -EV score with respect to β_{F_1-EV} . © 2024 IEEE

PCC between the F_1 -EV score and AUC-ROC decreases. This verifies once more that both performance metrics are indeed different. Furthermore, the PCC between the F_1 -EV score and optimal F_1 score increases for $\beta_{F_1-EV} < 1$ and slightly decreases for $\beta_{F_1-EV} > 1$ but still has a high correlation. Last and most importantly, the PCC between the F_1 -EV score and submitted F_1 scores decreases for $\beta_{F_1-EV} > 0.2$. However, the decrease is to less degree than it is the case for the PCC with AUC-ROC and therefore is relatively stable. Using a relatively small value for β_{F_1-EV} such as setting $\beta_{F_1-EV} = 0.2$ seems to be a good choice to have a threshold-independent performance measure that is similar to AUC-ROC score but also includes the difficulty of estimating a good decision threshold.

4.4 SUMMARY

In this chapter, it was investigated how to choose a good decision threshold for semi-supervised ASD systems. For this purpose, the threshold estimation methods presented in Section 2.11 were compared experimentally. It was shown that most methods perform equally well and yield a performance between 90% to 95% of the theoretically optimal performance. More detailed comparisons revealed that MST approaches perform slightly better than non-iterative approaches, especially when less data is used for training the system. Hence, MST yields more robust decision thresholds and thus is identified as the preferred threshold estimation method. Furthermore, it was shown that not using all normal training samples for training the embedding model with the goal of obtaining more realistic anomaly scores for estimating the threshold does not improve the performance. Hence, all normal samples that are available should be used for training the embedding model.

In a second section, it was shown that AUC-ROC does not include the difficulty of estimating a good decision threshold because a maximal AUC-ROC only shows that a single optimal decision threshold exists. Moreover, the AUC-ROC score has a low correlation with the F_1 score and thus only using the AUC-ROC score

for evaluating the performance of an [ASD](#) system is not sufficient. To solve this issue, the threshold-independent F_1 -[EV](#) was proposed, which has a high correlation with the [AUC-ROC](#) score as well as with the estimated and optimal F_1 scores. However, it was also shown that F_1 -[EV](#) requires properly defined bounds to yield meaningful results. Still, F_1 -[EV](#) has a strong potential to replace the [AUC-ROC](#) as the standard evaluation metric for semi-supervised [ASD](#).

A *domain shift* is a change of the underlying distribution of the data. There are several reasons for such a domain shift to occur. Examples are changing locations of acoustic sensors, changing the time at which data is collected, adding or removing other sound sources or modifying the monitored sound sources. The subset of the data space that has initially been of interest is called *source domain* $X_{\text{source}} \subset X$ and the domain shifted subset is called *target domain* $X_{\text{target}} \subset X$. Usually, there are only very few training samples available for the target domain, i.e. $|X_{\text{target}} \cap X_{\text{train}}| \ll |X_{\text{source}} \cap X_{\text{train}}|$. Due to this data scarcity, it is difficult to estimate the distribution of the normal data in the target domain. To still have a well-defined ASD task, one needs to make sure that shifting the domain of normal data still results in normal data and that anomalous data is mapped onto anomalous data. Thus, in the context of this thesis a domain shift is formally defined as any mapping $\text{shift}_{\text{domain}} : X_{\text{source}} \rightarrow X_{\text{target}}$ such that $\text{shift}_{\text{domain}}(x_n) \in X_{\text{normal}}$ and $\text{shift}_{\text{domain}}(x_a) \in X_{\text{anomalous}}$ for all $x_n \in X_{\text{normal}}$ and all $x_a \in X_{\text{anomalous}}$.

The main difficulty of handling domain shifts is that normal and anomalous data belonging to the target domain may both be viewed as weakly anomalous by a system trained on the source domain due to a mismatch between the underlying distributions of both domains. Note that, depending on the application, precisely defining the sets X_{source} and X_{target} may be very challenging. This is the reason why the formal definition of a domain shift as presented above is rather vague and will not be further specified. In the context of acoustic machine condition monitoring, which remains to be the application considered in this chapter, a domain shift corresponds to changing the acoustic environment, e.g. modifying the background noise or changing any parameter settings of fully-functioning machines such as the speed they are operating with.

Modifying an ASD system trained on the source domain to perform well on the target domain for which only very limited training data is available, is called *domain adaption* [96]. Ideally, this enables users to adapt the system without too much effort to react to possible changes of the acoustic environment or the monitored sound sources themselves. A possible method for domain adaption is to estimate different distributions for each domain using a jointly trained embedding space and use different backends to decide whether a sample is normal or anomalous [240]. However, adapting a system for each possible domain shift is highly impractical and costly because it requires trained personnel to collect at least some additional data belonging to the target domain, fine-tune parameters and re-train or even change entire components of an ASD system. Furthermore, normal data of the original source domain will likely not be considered normal after adapting the system to the target domain. It would be much better if the same system yields

reasonable performance regardless of domain shifts without needing to adapt the system. Achieving this goal of developing a domain-independent system is called *domain generalization* [225]. Hence, the main difference to domain adaptation is that the same models and the same backend with the same decision threshold are used for the source and all possible target domains. This makes domain generalization inherently more difficult but solving this problem automatically solves all domain adaptation problems.

To learn embeddings that are robust to domain shifts, [225] distinguishes between the two main categories *domain-invariant representation learning* [15] and *feature disentanglement* [271]. The idea of domain-invariant learning is to explicitly reduce the variability between multiple different source domains in the embedding space as this will also reduce the variability to arbitrary target domains. Disentangled feature learning aims at learning embeddings as combinations of domain-invariant features, which are robust to domain shifts, and domain-dependent features, which capture the variability between different domains. In [46], these two main categories were described as a *domain-mixing-based approach*, where a domain-invariant model is used for both domains [10, 219, 243], and a *domain-classification-based approach* with different models for the source and target domain, whose domain-specific anomaly scores are combined appropriately. Evidence has been provided that using a combination of domain-dependent models or distributions [269] or using a classifier for the domains [115] leads to better performance than a domain-mixing approach. The most likely reason is that a domain-invariant training requires to remove at least some potential classes and thus simplifies the classification task, which leads to less informative embeddings. Note that classifying between different meta information, as done in [218], corresponds to a disentangled feature learning approach because each combination of provided meta information that defines a class may also define a specific domain shift.

The goal of this chapter is to present an embedding-based ASD system that reliably detects anomalous sounds regardless of the domain a given audio recording belongs to. As this task is much more difficult than an ASD task without domain shifts, re-using the system presented in Chapter 3 alone will not suffice and thus the system must be modified accordingly. To this end, several techniques for improving the performance will be presented.

This chapter is structured as follows: First, two datasets used for the experimental evaluations in domain-shifted conditions will be presented. Then, an ASD system specifically focusing on domain generalization will be designed and individual design choices will be experimentally evaluated. The decisions obtained with differently trained systems will be explained in the third section by visualizing the influence of specific regions of the input samples and the embedding space. In the fourth section, the performance obtained with this ASD system is compared to systems based on pre-trained embeddings. The loss function AdaProj, which is a generalization of the sub-cluster AdaCos loss, will be presented in the fourth

section. Then, it will be investigated whether utilizing [SSL](#) improves the performance of an [ASD](#) system. The chapter is concluded by combining all techniques of the previous sections into a single system and comparing its performance to the state-of-the-art systems.

5.1 CONTRIBUTIONS OF THE AUTHOR

The sections of this chapter are largely based on the following key publications:

- Kevin Wilkinghoff. “Design Choices for Learning Embeddings from Auxiliary Tasks for Domain Generalization in Anomalous Sound Detection.” In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2023. DOI: [10.1109/ICASSP49357.2023.10097176](https://doi.org/10.1109/ICASSP49357.2023.10097176).
- Kevin Wilkinghoff and Frank Kurth. “Why do Angular Margin Losses work well for Semi-Supervised Anomalous Sound Detection?” In: *IEEE/ACM Transactions on Audio, Speech and Language Processing* 32 (2024), pp. 608–622. DOI: [10.1109/TASLP.2023.3337153](https://doi.org/10.1109/TASLP.2023.3337153)
- Kevin Wilkinghoff and Fabian Fritz. “On Using Pre-Trained Embeddings for Detecting Anomalous Sounds with Limited Training Data.” In: *31st European Signal Processing Conference*. IEEE, 2023, pp. 186–190. DOI: [10.23919/EUSIPCO58844.2023.10290003](https://doi.org/10.23919/EUSIPCO58844.2023.10290003).
- Kevin Wilkinghoff. “AdaProj: Adaptively Scaled Angular Margin Subspace Projections for Anomalous Sound Detection with Auxiliary Classification Tasks.” Submitted to 9th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE), arXiv:2403.14179. 2024. DOI: [10.48550/arXiv.2403.14179](https://doi.org/10.48550/arXiv.2403.14179).
- Kevin Wilkinghoff. “Self-Supervised Learning for Anomalous Sound Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 276–280. DOI: [10.1109/ICASSP48485.2024.10447156](https://doi.org/10.1109/ICASSP48485.2024.10447156).

For publications that are not single-authored, individual contributions of the thesis author and all co-authors to these publications are stated in [Section A.1](#). If not stated otherwise, the content listed in the following paragraph is the sole contribution of the thesis author.

The contents of the following sections were published as papers by the thesis author: [Section 5.3.2](#) is based on [\[245\]](#), [5.4](#) is based on [\[262\]](#), [Section 5.5](#) is based on [\[258\]](#), [Section 5.6](#) is based on [\[248\]](#) and [Section 5.7](#) is based on [\[250\]](#). Frank Kurth proposed to include [Table 21](#). Fabian Fritz helped with the experimental evaluations in [Section 5.5](#) by writing wrapper functions for the pre-trained embeddings. The experimental results presented in [Section 5.8](#) are an extended version of Fig.

Table 12: Structure of the DCASE2022 [ASD](#) dataset. The dataset contains recordings of 7 machine types and is divided into sections, which contain recordings of the same machine type and are used for evaluation.

subset	number of sections (per machine type)	split	number of recordings (per section)			
			source domain		target domain	
			normal	anomalous	normal	anomalous
development set	3	training	990	0	10	0
		test	50	50	50	50
evaluation set	3	training	990	0	10	0
		test	100 normal and 100 anomalous samples			

3 from [248], which also includes results using the [SSL](#) approaches presented in [Section 5.6](#).

5.2 MACHINE CONDITION MONITORING IN DOMAIN-SHIFTED CONDITIONS

All experiments in this chapter related to domain generalization for [ASD](#) use the following datasets for acoustic machine condition monitoring.

5.2.1 DCASE2022 [ASD](#) dataset

[Table 12](#) contains a summary of the DCASE2022 [ASD](#) dataset [46]. Overall, the dataset has a similar structure as the DCASE2020 dataset described in [Section 3.2.1](#) but also includes the difficulty of domain generalization. The dataset contains recordings of seven different machine types, namely “ToyCar” and “ToyTrain” from ToyAdmos2 [74] and “fan”, “gearbox”, “bearing”, “slide rail” and “valve” from MIMII-DG [45]. All recordings have a length of 10s and a sampling rate of 16 kHz. For each machine type, there are six different sections, each corresponding to a subset of the dataset with one specific domain shift, that are used to compute the performance. Thus, there are 42 sections in total. It is important to emphasize that, in contrast to the DCASE2020 dataset, the sections do not correspond to individual machines but may contain a mixture of several machines of a known type. This makes the dataset more realistic because it should be possible to use the same [ASD](#) system for multiple machines. However, the task is also more difficult because the variability between different recordings contained in the same section is increased.

The dataset is divided into a development and an evaluation split, each containing three sections of each machine type. Both splits consist of a training subset containing only normal training samples and a test split containing normal and

Table 13: Structure of the DCASE2023 [ASD](#) dataset. The dataset contains recordings of 14 machine types and is divided into sections, which contain recordings of the same machine type and are used for evaluation.

subset	number of sections (per machine type)	split	number of recordings (per section)			
			source domain		target domain	
			normal	anomalous	normal	anomalous
development set	1	training	990	0	10	0
		test	50	50	50	50
evaluation set	1	training	990	0	10	0
		test	200 samples			

anomalous samples. Furthermore, each section is divided into a source domain with 990 normal training samples and a target domain that differs from the source domain by modifying machine parameter settings or the noise conditions and for which only ten normal training samples are available. In addition to the section and machine type, so-called *attribute information* are provided for each normal training sample as for example the speed of a toy car, different machine IDs or noise levels. For all files of both test splits, only machine types and sections are provided. Combining machine types, sections and attribute values present in the development and evaluation split of the dataset into a single label, which thus encodes the meta information for each file, results in 342 classes.

The performance obtained on this dataset is measured by calculating the [AUC-ROC](#) and [pAUC](#) with $p = 0.1$ for each section and computing the harmonic mean over all individual performance metrics. Using the harmonic mean instead of the arithmetic mean ensures that bad scores penalize the total performance result more severely to favor [ASD](#) systems that perform equally well for all sections, which is favorable for practical applications. The [AUC-ROCs](#) and [pAUCs](#) are calculated independently of the domain, i.e. using only a single [ROC](#) curve for both domains.

5.2.2 DCASE2023 [ASD](#) dataset

The DCASE2023 [ASD](#) dataset [44] summarized in [Table 13](#) is similar to the DCASE2022 [ASD](#) dataset but differs in the following aspects: The major difference is that the DCASE2023 dataset is designed for so-called *first-shot ASD* meaning that there are different machine types for the development and evaluation split. This makes it impossible to fine-tune the hyperparameters of the system to individual machine types by evaluating the performance, which indirectly uses the anomalous samples of the development set as additional training samples and thus should not be allowed in a semi-supervised anomaly detection setting. For

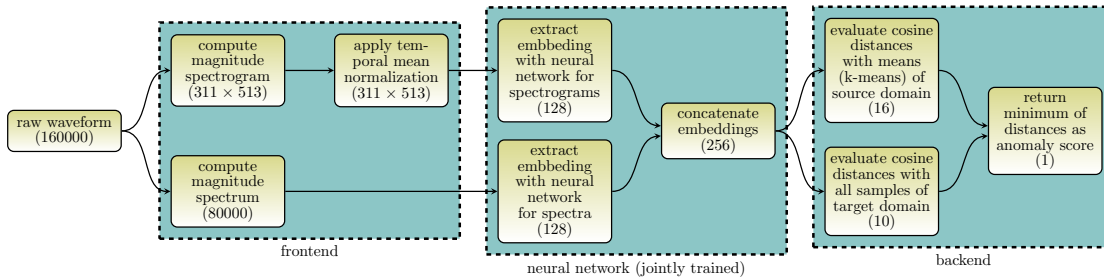


Figure 21: Structure of the proposed anomalous sound detection system for domain-shifted conditions. Representation size in each step is given in brackets. © 2023 IEEE

the development split, the same machine types as for the DCASE2022 dataset are used. For the evaluation split, the machine types “ToyTank”, “ToyNscale” and “ToyDrone” from ToyAdmos2+ [73] and “vacuum”, “bandsaw”, “grinder” and “shaker” from MIMII-DG [45] are used. A second major change is that there is only a single section for each machine type. This increases the difficulty by making the classification task imposed by discriminating between provided meta information less challenging and thus resulting in less informative embeddings. As a third increase in difficulty, the length of the recordings belonging to different machine types varies between 6 s and 18 s.

5.3 DESIGNING AN ASD SYSTEM FOR DOMAIN GENERALIZATION

In this section, an ASD system specifically designed to work well in domain-shifted conditions will be presented and evaluated. The main idea is to use a single domain-independent embedding model that extracts embeddings of the source and target domains. Then, domain-dependent distances to normal data samples can be combined into a single domain-independent anomaly score yielding similar performance for both domains.

5.3.1 System description

The ASD system presented in Chapter 3 yields state-of-the-art performance for ASD without domain shifts and thus serves as a basis for developing an ASD system in domain-shifted conditions. A modified version is depicted in Figure 21. As before, the system consists of three main blocks, namely a frontend for computing feature representations, a neural network for extracting embeddings and a backend for computing anomaly scores. All three blocks will now be presented in detail. Individual design choices will be investigated experimentally in the following subsection using the DCASE2022 dataset.

The frontend uses two branches extracting different feature representations of the raw waveforms: A short-time Fourier transform (STFT) and a discrete Fourier

Table 14: Sub-network architecture for DFT feature branch. © 2023 IEEE

layer name	structure	output size
input	-	80000
1D convolution	256, stride= 64	1250×128
1D convolution	64, stride= 16	40×128
1D convolution	16, stride= 4	10×128
flatten	-	1280
dense	-	128
dense	-	128
dense	-	128
dense	-	128
dense (embedding)	no activation	128

transform (DFT) based feature. For the STFT based feature branch, magnitude spectrograms extracted with Hanning-weighted windows with a size of 1024 and a hop size of 512 are extracted. In contrast to the system presented in Section 3.2.2, no Mel filterbank is used and no logarithm is applied. The spectrograms are further pre-processed with temporal mean normalization (TMN), which essentially removes all constant frequency information while also denoising the data similar to cepstral mean normalization [192]. The main reason to apply TMN is to make both feature representations more different. For the second feature branch, DFTs are computed to capture all constant frequencies with the highest possible frequency resolution. Since machine sounds are very structured with highly repetitive patterns, utilizing as much frequency information as possible may be beneficial to detect anomalous sounds.

The neural network used to learn an embedding space utilizes another sub-network for each feature branch: A one-dimensional CNN for the DFT features (see Table 14) and a modified version of the ResNet architecture shown in Table 4 (see Table 15). The learned embeddings of both sub-networks are concatenated into a single embedding. In contrast to taking a weighted sum of both embeddings or combining them with a trainable layer, the network has no possibility to only utilize a single embedding that is most useful for solving the auxiliary task and ignore the other embedding. This ensures a higher sensitivity of the system for detecting anomalous data as anomalies may only be detectable for one of the input features. In each convolutional or dense layer of both sub-networks, batch normalization [86] and the activation function ReLU are used. The entire network is jointly trained for ten epochs with a batch size of 64 by minimizing the sub-cluster AdaCos loss with 16 sub-clusters for each class using adam [101]. Mixup [272] with a uniformly sampled mixing coefficient is used to randomly augment the

Table 15: Modified ResNet architecture for [STFT](#) feature branch. © 2023 IEEE

layer name	structure	output size
input	TMN	311×513
2D convolution	7×7 , stride= 2	$156 \times 257 \times 16$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$77 \times 128 \times 16$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$39 \times 64 \times 32$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$20 \times 32 \times 64$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$10 \times 16 \times 128$
max pooling	10×1 , stride= 1	$1 \times 16 \times 128$
flatten	-	2056
dense (embedding)	no activation	128

data during training. The classes used for training the model consist of all possible combinations of machine types, sections and different values for provided attribute information present in the training datasets of the development and evaluation set resulting in a total of 342 classes for the DCASE2022 dataset. Note that this training paradigm corresponds to a *feature disentanglement* approach for learning embeddings that are robust to domain-shifts. It is also possible to use multiple losses for different classification tasks instead of a single one [218, 240]. However, this did not lead to a noticeable difference in performance and thus only a single classification task is used for the sake of simplicity. Since it is possible that the network learns trivial solutions for a few classes that are very easy to identify, the strategies to prevent one-class models from learning trivial solutions are applied. More concretely, this means that no bounded activation functions, no trainable class centers, and no bias terms are used for both sub-networks. All centers are randomly initialized on the unit sphere and thus are pairwise orthogonal with very high probability.

The backend employs different strategies to compute an anomaly score in both domains. For the source domain, k-means with 16 clusters is applied to all normal training samples of the source domain. An anomaly score for the source domain is obtained by computing the minimum cosine distance to all resulting clusters. For the target domain, the minimum cosine distance to all normal training samples of the target domain is used as an anomaly score. As the final step, a domain-

Table 16: Comparison between using or not using trainable cluster centers and bias terms on the DCASE2022 dataset. © 2023 IEEE

dataset	domain	trainable cluster centers		non-trainable cluster centers	
		AUC-ROC (%)	pAUC (%)	AUC-ROC (%)	pAUC (%)
using bias terms					
dev	source	81.65 ± 0.85	70.25 ± 1.31	82.75 ± 0.95	75.22 ± 0.78
dev	target	77.18 ± 0.88	62.05 ± 1.10	76.84 ± 1.39	61.66 ± 1.09
dev	mixed	77.73 ± 0.90	64.09 ± 1.40	79.79 ± 0.50	64.89 ± 0.41
eval	source	74.21 ± 1.14	62.84 ± 1.38	76.45 ± 0.71	65.12 ± 0.66
eval	target	71.13 ± 1.24	59.54 ± 0.86	69.19 ± 0.56	59.08 ± 1.15
eval	mixed	71.91 ± 0.98	58.84 ± 0.88	73.04 ± 0.58	59.18 ± 0.90
not using bias terms					
dev	source	82.88 ± 0.68	71.44 ± 0.96	84.19 ± 0.75	76.45 ± 0.90
dev	target	77.68 ± 1.11	62.82 ± 1.17	78.51 ± 0.90	62.54 ± 0.87
dev	mixed	78.15 ± 0.77	64.86 ± 0.52	81.36 ± 0.66	66.55 ± 0.86
eval	source	74.34 ± 0.96	63.49 ± 0.38	76.81 ± 0.79	65.84 ± 0.22
eval	target	71.30 ± 0.33	59.94 ± 0.81	69.80 ± 0.53	59.67 ± 1.14
eval	mixed	72.15 ± 0.50	58.90 ± 0.42	73.43 ± 0.54	59.78 ± 0.83

independent anomaly score is given by the minimum of both domain-specific anomaly scores.

5.3.2 Experimental investigations of individual design choices

To show that the design choices of the proposed ASD system actually improve the performance, multiple experiments have been conducted and will now be discussed. Each experiment was repeated five times and the mean and standard deviation of the harmonic means computed over all sections belonging to the development and evaluation split of the DCASE2022 dataset are shown.

In the first experiment it is evaluated whether applying the same strategies used to prevent the embedding model to learn trivial solutions when using one-class losses also has a positive impact on the performance when using an angular margin loss. The results presented in Table 16 show that the overall performance is significantly improved when not using bias terms and not adapting the centers during training. It is worth noting that the performance improves on the target domain but degrades on the source domain when using trainable centers. This indicates that for some classes of the source domain the embedding model indeed

Table 17: Comparison between different input feature representations and ways of combining them on the DCASE2022 dataset. © 2023 IEEE

		individual input feature representations					
dataset	domain	magnitude spectrum		log-Mel magnitude spectrogram		magnitude spectrogram	
		AUC-ROC (%)	pAUC (%)	AUC-ROC (%)	pAUC (%)	AUC-ROC (%)	pAUC (%)
dev	source	80.75 ± 0.92	71.09 ± 1.14	70.91 ± 2.10	63.21 ± 1.10	79.18 ± 1.07	70.38 ± 0.43
dev	target	73.95 ± 1.25	61.31 ± 1.40	65.78 ± 1.48	57.05 ± 0.76	76.59 ± 1.59	60.59 ± 1.50
dev	mixed	76.81 ± 0.94	63.09 ± 1.23	68.93 ± 1.52	57.79 ± 0.55	77.60 ± 1.02	62.31 ± 1.10
eval	source	68.42 ± 1.02	59.06 ± 0.89	67.59 ± 1.01	59.96 ± 0.65	74.65 ± 0.83	64.04 ± 1.23
eval	target	63.46 ± 1.39	56.90 ± 0.95	62.09 ± 0.61	56.60 ± 0.79	69.67 ± 1.14	58.95 ± 0.74
eval	mixed	66.00 ± 1.11	57.11 ± 0.58	65.04 ± 0.68	57.00 ± 0.50	72.10 ± 0.81	58.87 ± 0.33
		combining magnitude spectrum and magnitude spectrograms					
dataset	domain	concatenate embeddings after training		add embeddings while training		concatenate embeddings while training	
		AUC-ROC (%)	pAUC (%)	AUC-ROC (%)	pAUC (%)	AUC-ROC (%)	pAUC (%)
dev	source	84.68 ± 0.86	74.51 ± 0.26	83.12 ± 1.18	73.25 ± 1.11	84.19 ± 0.75	76.45 ± 0.90
dev	target	78.78 ± 0.78	63.00 ± 0.36	77.96 ± 1.37	62.02 ± 1.11	78.51 ± 0.90	62.54 ± 0.87
dev	mixed	81.17 ± 0.66	65.23 ± 0.39	80.21 ± 0.73	64.40 ± 1.20	81.36 ± 0.66	66.55 ± 0.86
eval	source	75.27 ± 0.96	63.93 ± 0.63	75.54 ± 0.83	64.85 ± 0.73	76.81 ± 0.79	65.84 ± 0.22
eval	target	69.31 ± 0.90	59.45 ± 0.67	69.71 ± 0.39	59.08 ± 1.15	69.80 ± 0.53	59.67 ± 1.14
eval	mixed	72.18 ± 0.67	59.45 ± 0.46	72.48 ± 0.53	59.22 ± 0.66	73.43 ± 0.54	59.78 ± 0.83

learns solutions that are closely resembling trivial solutions making it difficult to discriminate between normal and anomalous embeddings.

In Table 17, different input feature representations and approaches for combining them are compared. The following observations can be made: First, magnitude spectrograms perform much better than log-Mel magnitude spectrograms that are used in many other ASD systems (cf. Section 2.2). The most likely reason is that high frequencies are more important than low frequencies for detecting anomalous sounds of machines [140] and using the logarithmically scaled Mel-filterbank decreases the resolution for high frequencies. Furthermore, only using the DFT-based magnitude spectra leads to surprisingly good results that are better than the ones obtained with log-Mel spectrograms but worse than when using spectrograms. Hence, a high frequency resolution is highly beneficial to detect anomalous sounds for machine condition monitoring. Not surprisingly, combining both feature branches improves the performance as providing different views on the data gives a more complete picture of the input data. Thirdly, concatenating the embeddings of the sub-networks while training leads to the best performance. Again, the reason is that the network is forced to encode as much information as possible for each individual feature branch and thus results in more informative embeddings.

In Table 18, it is experimentally verified whether applying TMN to the magnitude spectrograms improves performance or not. It can be seen that applying TMN leads to very similar performance when only using the magnitude spectrograms. But, when utilizing both feature branches there is an improvement in performance.

Table 18: Effect of temporal normalization in domain-shifted conditions. © 2023 IEEE

dataset	domain	without temporal normalization		with temporal normalization	
		AUC-ROC (%)	pAUC (%)	AUC-ROC (%)	pAUC (%)
magnitude spectrogram					
dev	source	79.93 ± 0.71	70.98 ± 1.38	79.18 ± 1.07	70.38 ± 0.43
dev	target	76.18 ± 0.85	60.35 ± 1.23	76.59 ± 1.59	60.59 ± 1.50
dev	mixed	77.69 ± 0.47	62.52 ± 1.09	77.60 ± 1.02	62.31 ± 1.10
eval	source	75.53 ± 1.19	64.31 ± 1.08	74.65 ± 0.83	64.04 ± 1.23
eval	target	69.52 ± 0.73	59.67 ± 0.71	69.67 ± 1.14	58.95 ± 0.74
eval	mixed	72.19 ± 0.77	59.39 ± 0.91	72.10 ± 0.81	58.87 ± 0.33
magnitude spectrum + magnitude spectrogram					
dev	source	83.18 ± 1.68	74.66 ± 0.71	84.19 ± 0.75	76.45 ± 0.90
dev	target	76.95 ± 0.97	62.26 ± 0.62	78.51 ± 0.90	62.54 ± 0.87
dev	mixed	80.04 ± 0.76	64.84 ± 0.51	81.36 ± 0.66	66.55 ± 0.86
eval	source	76.41 ± 0.48	65.39 ± 0.68	76.81 ± 0.79	65.84 ± 0.22
eval	target	68.89 ± 0.96	59.46 ± 0.68	69.80 ± 0.53	59.67 ± 1.14
eval	mixed	72.85 ± 0.61	59.91 ± 0.75	73.43 ± 0.54	59.78 ± 0.83

The reason is that both features better complement each other as the constant frequency information is removed from the spectrograms, which is exactly the information being captured by the [DFT](#) features, and thus both features provide a more detailed view on the data than when not applying [TMN](#).

In [Table 19](#), the performances obtained with different backends for computing anomaly scores are compared. More concretely, using a [GMM](#) is compared to using the cosine distance and different ways of combining the anomaly scores of both domains are evaluated. The following observations can be made. Unsurprisingly, using only domain-specific anomaly scores leads to the best performance on the domain they belong to. When comparing ways of computing anomaly scores, a domain-specific [GMM](#) leads to better performance than cosine distance on the source domain, which is consistent with the findings presented in [Section 3.4.3](#). For the target domain, both approaches are in fact equivalent due to the small number of training samples and thus lead to the exact same performance. However, the focus of this chapter lies on generalizing to unseen domains while still performing well on the source domain without the need of adapting the model. This requires a single decision threshold for both domains. When investigating the performance on the mixed domain, a joint model performs much better than individual models or when using the sum of the individual scores. Interestingly, the joint model even outperforms the sub-model specialized on the source domain showing once more that some classes are too easily detected by this model, which leads to embeddings without sufficient information to detect anomalies. Although a joint model in combination with a [GMM](#) leads to better performance on the source domain than when using cosine distance, the cosine distance performs better on the target domain and also better in the mixed domain. As optimizing the domain-independent performance is the aim of this chapter, using cosine similarity and training a joint model is the preferred design choice.

Lastly, the optimized [ASD](#) system investigated in this section and the baseline system presented in [Section 3.2.2](#) are compared in [Table 20](#). One can see that all performance improvements resulting from the modifications add up and lead to a strong difference in performance, clearly favoring the modified system in domain-shifted conditions.

5.4 EXPLAINING THE DECISIONS

The goal of this section is to explain the decisions of the [ASD](#) system by visualizing the obtained results. Explaining the decisions of data-driven models is important for practical applications to increase the acceptance and trust in the results (explainable artificial intelligence ([xAI](#)) [[84](#)]). It also provides the possibility to detect potential errors and gain additional insights about possible reasons that caused these errors. For acoustic machine condition monitoring, localizing anomalous temporal regions or frequency bands is crucial to help users to find the cause of mechanical failure and thus simplifies the maintenance process. Previous work

Table 19: Comparison between different backends on the DCASE2022 dataset. © 2023 IEEE

dataset	domain	GMM		cosine distance	
		AUC-ROC (%)	pAUC (%)	AUC-ROC (%)	pAUC (%)
using scores from source domain model only					
dev	source	82.97 ± 0.97	77.36 ± 0.38	83.10 ± 1.02	76.87 ± 0.26
dev	target	66.52 ± 0.42	59.63 ± 0.70	71.66 ± 1.25	61.45 ± 0.83
dev	mixed	71.48 ± 0.26	58.86 ± 0.38	76.72 ± 0.78	63.37 ± 0.71
eval	source	77.46 ± 1.16	66.73 ± 0.56	76.68 ± 0.85	66.25 ± 0.51
eval	target	44.23 ± 3.67	54.87 ± 0.46	57.90 ± 1.17	55.62 ± 1.24
eval	mixed	63.83 ± 0.62	55.74 ± 0.33	67.24 ± 0.70	56.83 ± 0.92
using scores from target domain model only					
dev	source	62.41 ± 2.80	60.54 ± 1.44	62.42 ± 2.80	60.55 ± 1.44
dev	target	79.93 ± 0.92	62.19 ± 1.05	79.92 ± 0.92	62.18 ± 1.06
dev	mixed	70.84 ± 1.13	58.36 ± 1.38	70.84 ± 1.13	58.36 ± 1.38
eval	source	52.82 ± 3.40	56.27 ± 1.17	52.80 ± 3.41	52.26 ± 1.17
eval	target	71.15 ± 0.50	60.72 ± 0.95	71.15 ± 0.50	60.72 ± 0.95
eval	mixed	62.55 ± 0.79	54.66 ± 0.92	62.55 ± 0.79	54.65 ± 0.92
using sum of scores from both domain models					
dev	source	75.94 ± 2.70	70.34 ± 2.69	79.44 ± 1.95	73.29 ± 2.60
dev	target	78.64 ± 1.12	63.28 ± 0.94	78.84 ± 1.23	62.67 ± 0.98
dev	mixed	77.10 ± 1.59	65.12 ± 1.63	77.83 ± 1.56	66.11 ± 1.61
eval	source	64.57 ± 1.35	60.96 ± 1.21	68.96 ± 1.11	63.09 ± 0.87
eval	target	66.69 ± 0.54	58.73 ± 0.97	67.84 ± 0.48	58.80 ± 1.33
eval	mixed	65.70 ± 0.91	57.26 ± 0.77	67.98 ± 0.64	58.12 ± 0.71
using scores from joint model for both domains					
dev	source	84.57 ± 0.70	77.57 ± 0.41	84.19 ± 0.75	76.45 ± 0.90
dev	target	77.26 ± 1.02	63.12 ± 1.24	78.51 ± 0.90	62.54 ± 0.87
dev	mixed	80.06 ± 0.35	64.67 ± 0.70	81.36 ± 0.66	66.55 ± 0.86
eval	source	78.15 ± 0.95	67.55 ± 0.63	76.81 ± 0.79	65.84 ± 0.22
eval	target	65.17 ± 0.48	58.59 ± 0.79	69.80 ± 0.53	59.67 ± 1.14
eval	mixed	71.35 ± 0.53	59.32 ± 0.83	73.43 ± 0.54	59.78 ± 0.83

Table 20: Effect of the presented design choices for improving the performance in domain-shifted conditions. © 2023 IEEE

dataset	domain	standard design choices		presented design choices	
		AUC-ROC (%)	pAUC (%)	AUC-ROC (%)	pAUC (%)
dev	source	71.65 ± 1.07	62.71 ± 0.74	84.19 ± 0.75	76.45 ± 0.90
dev	target	63.63 ± 1.11	55.28 ± 0.84	78.51 ± 0.90	62.54 ± 0.87
dev	mixed	67.72 ± 0.79	55.94 ± 0.54	81.36 ± 0.66	66.55 ± 0.86
eval	source	69.11 ± 1.05	58.46 ± 0.42	76.81 ± 0.79	65.84 ± 0.22
eval	target	61.33 ± 0.70	54.82 ± 0.78	69.80 ± 0.53	59.67 ± 1.14
eval	mixed	64.59 ± 0.74	55.22 ± 0.66	73.43 ± 0.54	59.78 ± 0.83

on explaining the decision of ASD systems for acoustic machine condition monitoring exists but is rather limited. In [50], uniform manifold approximation and projection (UMAP) [144] was used to visualize stacked consecutive frames of log(-Mel) magnitude spectrograms or openL3 embeddings [32] as dimension-reduced representations in a vector space. In [140], it has been shown that high frequency information is much more important than low frequency information for detecting anomalous machine sounds by using local interpretable model-agnostic explanations (LIME) [191] applied to sounds (SLIME) [151]. This verifies the choice of applying a high-pass filter to the input data representations (cf. Section 3.2.2).

5.4.1 Visualizing the input as viewed by the model

To visualize how using an auxiliary classification task affects the decision making of the ASD system based on specific regions, *randomized input sampling for explanation (RISE)* [181] is used. RISE applies random binary masks to the input and evaluates the performance with the masked input. Then, so-called *importance maps* are generated by repeating both steps many times for the same input sample and summing all masks weighted with the corresponding performance. Finally, the importance maps are normalized with the expected value of a random binary mask. As a result, such an importance map visualizes the importance of specific regions on the resulting performance. For the experiments conducted in this section, only the sub-model based on magnitude spectrograms of the ASD system described in Section 5.3.2 is used because these time-frequency representations allow to detect and visualize temporal patterns and frequency bands of interest.

The spectrograms used by the ASD system have a very high dimension with a temporal dimension of $D_{\text{time}} = 311$ and a frequency dimension of $D_{\text{freq}} = 513$. Hence, there are $2^{\text{T}\cdot\text{F}} = 2^{159543}$ possible binary masks for these spectrograms and thus too many iterations for RISE would be required. To reduce the dimension of the search space, the following strategies were employed: First, the temporal

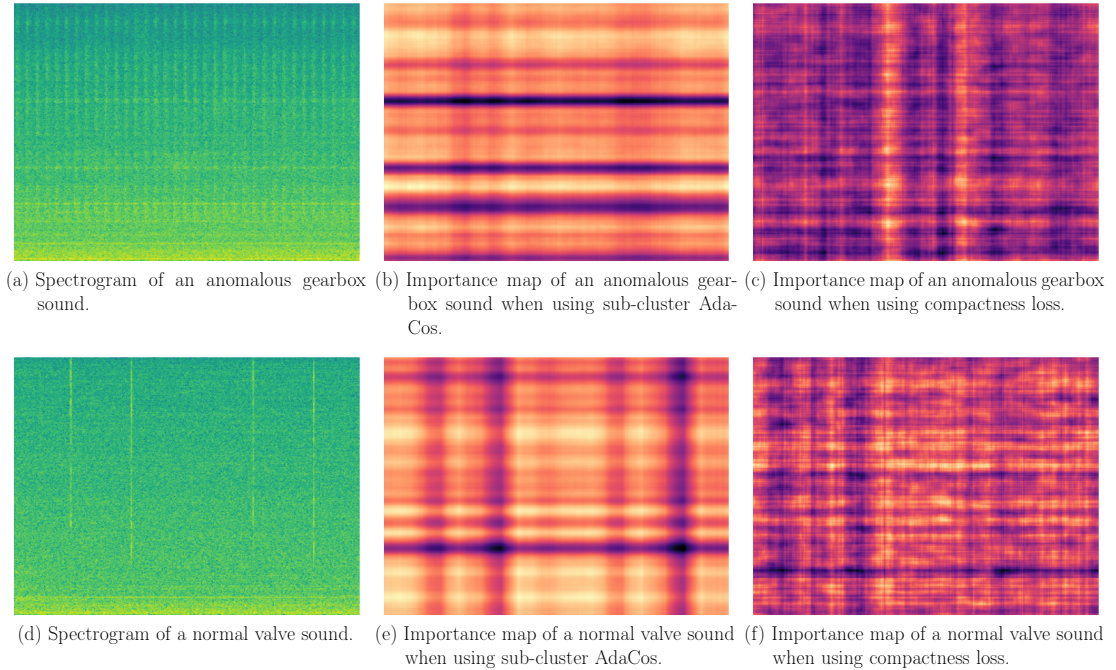


Figure 22: Log-scaled spectrograms (left column), importance maps obtained with **RISE** when training with the sub-cluster AdaCos loss and classifying between different machine types, sections and attribute information (middle column), and importance maps obtained with **RISE** when training with an intra-class compactness loss and without an auxiliary classification task (right column) for two different recordings belonging to the test split of the development set (rows). For the importance maps, blue colors indicate normal regions and yellow colors indicate regions that are found to be anomalous by the model. All subfigures use individual color scales to improve visual appearance for differently scaled importance maps and thus colors of different subfigures cannot be compared to each other. © 2024 IEEE

and frequency dimension were treated individually by masking individual time frames and frequency bands with a probability of 0.25 and combining both masks by multiplying them. This results in a reduction from $2^{T \cdot F}$ to 2^{T+F} possibilities. Note that for machine condition monitoring, such an approach is not too restrictive because machine sounds are highly structured, meaning that they are mostly constant over time (e.g. fans), are constant for some duration (e.g. slider rails) or consist of short sound events over a wide frequency range (e.g. valves). A further reduction of the search space was achieved by up-sampling and cropping small binary masks as also proposed for the original **RISE** algorithm [181]. For the time masks a size of 20 and for frequency masks a size of 34 were used, resulting in a search space of 2^{54} , which is still huge but much smaller than the original space. In all experiments, 640,000 iterations were used to generate a single importance map.

Figure 22 contains log-scaled magnitude spectrograms of two samples and importance maps resulting from a model trained by minimizing the sub-cluster AdaCos loss and from a model trained by minimizing an intra-class compactness loss. Note that the results of the ASD system are not perfect and thus specific regions as visualized by the model do not need to be correct. Moreover, there are only single binary labels available for each recording, stating whether the sample is normal or anomalous. Hence, there is no ground truth about specific regions of the spectrograms available for further inspection. Manual inspection is not economical as the author of this thesis does not have the required application-dependent expertise to do this. Still, for the purpose of comparing a model trained with an auxiliary classification task and a one-class model, these visualizations are sufficiently detailed. The results obtained for both samples will now be discussed in detail.

For the first sample, which is an anomalous sound of a gearbox, the model trained with an auxiliary classification task monitors specific frequency bands that are found to be normal (blue horizontal lines in Figure 22(b)). Interestingly, these monitored regions correspond to the frequency bands that contain high energy and thus can also be found in Figure 22(a). The frequency band that is considered to be most anomalous by this model is located between the bottom two normal frequency bands and contains only low energy. Therefore, a normal sound is expected to have more or even less energy there. In contrast, the one-class model whose importance map is depicted in Figure 22(c) does not seem to monitor specific frequencies. Moreover, there are two anomalous temporal locations present (vertical lines in yellow). These do not seem to correspond to specific sound events in the recording because, when comparing it to the spectrogram, there are no visually noticeable events at these temporal locations. Thus, these seem to be errors made by the one-class model.

Similar results can be seen for the second sample containing a normal valve sound. Here, four sound events of very short durations can be seen in the spectrogram depicted in Figure 22(d). The importance map of the model trained with an auxiliary classification task considers these events to be normal since there are four vertical blue lines present in Figure 22(e). However, these events are not clearly visible in the importance map of the one-class model as shown in Figure 22(f). Here, the importance map looks almost completely random except for a frequency band that is considered to be normal but does not correspond to any region of the spectrogram with higher energy. Overall, these results support the claim that using an auxiliary classification task leads to more meaningful embeddings as the importance maps are noticeably more structured than the ones obtained for a one-class model.

5.4.2 Visualizing the embedding space

Another approach to explain the results of the ASD system is to visually inspect the embedding space using t-distributed stochastic neighbor embeddings (t-SNEs).

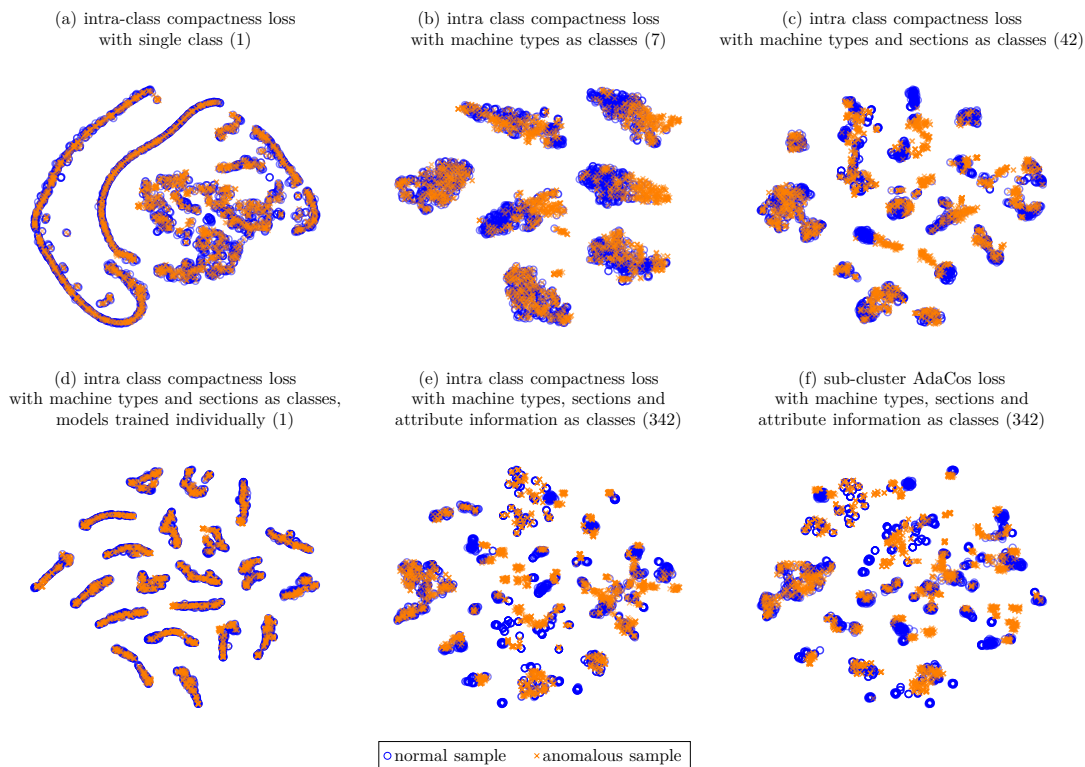


Figure 23: Visualizations of the test split of the development set in the learned embedding space for different loss functions and auxiliary tasks using t -SNE. Numbers in brackets denote the number of different classes used for the auxiliary task.
 © 2024 IEEE

Note that, according to Lemma 3.1, using Euclidean and cosine distance to measure the similarity of embeddings are equivalent approaches on the unit sphere and thus a standard implementation of t -SNE based on the Euclidean distance, as provided in scikit-learn [176], can be used. The results are depicted in Figure 23. Similar to the experimental results obtained on the DCASE2020 dataset that were presented in Section 3.3.3, it can be seen that using more classes helps to separate normal and anomalous samples (Figure 23(b), (c), (e) and (f)). In contrast, there is no visual difference between normal and anomalous samples when using a single or individual one-class losses (Figure 23(a) and (d)). It shall be noted that the model has also not learned a trivial solution as this would result in a uniformly distributed t -SNE embedding space. Therefore, the regularization techniques applied to the one-class losses prevented the embedding space from collapsing to a single point. All of these visual impressions are verified in Table 21 by measuring the distance between each anomalous sample and the closest normal sample. As stated in Chapter 3, the most likely reason for this significant difference in performance obtained with different loss functions is the same as before, namely the background noise. Teaching the model to discriminate between different classes enables it to closely monitor target machine sounds and more robustly detect de-

Table 21: Mean and standard deviation of the average Euclidean distance between the [t-SNE](#) projections of each anomalous sample and the closest normal sample over five trials for different losses and using different auxiliary tasks. © 2024 IEEE

loss	classes of auxiliary task (number of classes)	average distance
intra-class compactness loss	none (1)	0.485 ± 0.007
intra-class compactness loss	machine types (7)	1.636 ± 0.037
intra-class compactness loss	machine types and sections (42)	2.175 ± 0.075
intra-class compactness loss	machine types and sections, models trained individually (1)	0.559 ± 0.002
intra-class compactness loss	machine types, sections and attribute information (342)	2.646 ± 0.045
sub-cluster AdaCos loss	machine types, sections and attribute information (342)	2.947 ± 0.022

viations from normal behavior by mostly ignoring the background noise. As shown in [Figure 23](#), this is not the case for individual one-class losses.

5.5 COMPARISON TO PRE-TRAINED EMBEDDINGS

Without a sufficient amount of training data, using pre-trained embeddings is a promising approach for many closed-set classification tasks. Since there are only a few normal training samples available for the target domain, this motivates to compare the performance of the [ASD](#) system presented in the previous section to a system based on pre-trained embeddings. To this end, the four pre-trained embeddings [VGGish](#), [OpenL3](#), [PANN](#) and [Kumar](#) presented in [Section 2.5](#) will be evaluated.

5.5.1 System design for pre-trained embeddings

To utilize the pre-trained embeddings for [ASD](#), a shallow classifier is trained as proposed in [\[32, 67, 162\]](#). Similar to a directly trained model, this classifier predicts the correct meta information using the pre-trained embeddings as input. Before inserting the embeddings as input, they are standardized using batch normalization [\[86\]](#) as this has been shown to improve performance [\[32\]](#). The model consists of three layers with 512, 128 and 128 neurons. For the first two layers, [ReLU](#) is used as an activation function and batch normalization is applied. Prior to the last layer, dropout [\[81\]](#) is applied with a probability of 50%. The last layer consists of a linear transformation and a projection onto the hypersphere by using the sub-cluster AdaCos loss with 16 sub-clusters per class. The model is trained for 100 epochs with a batch size of 64 using Adam [\[101\]](#). As for the directly trained model, mixup [\[272\]](#) is applied, no bias terms are used and the randomly initialized centers are not adapted during training to avoid learning trivial solutions for easily recognizable classes. Anomaly scores are obtained by estimating the distribution of the normal training samples belonging to single sections of the dataset with a

Table 22: Harmonic means of **AUC-ROCs** obtained with different ways to handle the temporal dimension of pre-trained embeddings. © 2023 IEEE

dataset	embeddings	mean of embeddings			mean of scores	native
		before training	during training	after training		
dev set	VGGish	65.78 ± 0.37	64.98 ± 0.25	58.47 ± 0.58	59.27 ± 0.58	not available
dev set	OpenL3	70.94 ± 1.36	70.83 ± 0.93	59.85 ± 0.49	62.67 ± 1.36	not available
dev set	PANN	64.80 ± 0.25	66.30 ± 0.55	59.66 ± 0.32	60.47 ± 0.17	64.21 ± 0.17
dev set	Kumar	66.04 ± 0.76	65.85 ± 0.83	58.94 ± 1.00	62.22 ± 0.98	60.97 ± 0.52
eval set	VGGish	64.69 ± 0.34	63.91 ± 0.73	58.30 ± 0.98	59.78 ± 0.53	not available
eval set	OpenL3	69.06 ± 0.42	68.70 ± 0.94	62.44 ± 0.46	65.02 ± 1.04	not available
eval set	PANN	63.55 ± 0.27	65.29 ± 0.39	58.57 ± 1.07	60.34 ± 0.62	63.33 ± 0.36
eval set	Kumar	63.56 ± 0.59	64.05 ± 0.27	56.95 ± 0.86	61.04 ± 0.44	60.13 ± 0.24

GMM and computing the log-likelihood of individual test samples. For the openL3 embeddings, the model pre-trained on environmental sounds is used because these are more closely related to machine sounds than music.

5.5.2 Experimental results

In the following, multiple experiments are conducted to investigate how to design an **ASD** system based on pre-trained embeddings. The networks for extracting OpenL3 and VGGish embeddings use a sliding window resulting in embeddings with a temporal dimension, i.e. multiple vector-sized embeddings per sample. To obtain a single anomaly score for a given file, a proper strategy for handling all embeddings belonging to this file needs to be chosen. Possible choices are to compute the mean of the embeddings before, during or after training or to compute the mean of the individual anomaly scores. An experimental comparison of the performances obtained with these strategies can be found in Table 22. For PANN and Kumar embeddings, which do not have a temporal dimension, a sliding window with a length of 960 ms is used to artificially create a temporal dimension allowing to also evaluate the same strategies for these embeddings. In contrast to the results presented in [67], the best results are obtained when combining the embeddings before or during training and not after training, which led to significantly worse results. Interestingly, artificially creating a time-dimension for PANN and Kumar embeddings increases the performance. This shows that some information important for detecting anomalous samples is lost when using the pre-trained models to full extent.

As discussed in Section 2.6 and experimentally investigated for directly trained **ASD** systems, there are several possible backends for computing an anomaly score. One can directly estimate the distribution with a **GMM**, apply **PCA** or **LDA** before estimating the distribution or train a shallow classifier using different loss functions. In Table 23, the performances obtained with different backends are

Table 23: Harmonic means of **AUC-ROCs** for different backends and considered embeddings. Only deviations from the standard system are stated in the header of the table. © 2023 IEEE

dataset	embeddings	without trained shallow classifier			with trained shallow classifier		
		-	PCA	LDA	-	CXE as loss	cosine distance as backend
dev set	VGGish	60.22 ± 0.25	60.25 ± 0.43	62.90 ± 0.13	64.45 ± 0.60	65.40 ± 0.55	65.78 ± 0.37
dev set	OpenL3	66.82 ± 0.19	66.33 ± 0.12	64.66 ± 0.24	67.83 ± 1.24	68.99 ± 0.61	70.94 ± 1.36
dev set	PANN	60.36 ± 0.09	61.48 ± 0.18	60.09 ± 0.45	64.39 ± 0.75	63.82 ± 0.56	66.30 ± 0.55
dev set	Kumar	61.47 ± 0.26	61.92 ± 0.29	61.82 ± 0.26	64.22 ± 0.63	64.08 ± 1.54	65.85 ± 0.83
eval set	VGGish	57.48 ± 0.36	57.47 ± 0.18	61.47 ± 0.20	62.00 ± 1.26	63.77 ± 0.63	64.69 ± 0.34
eval set	OpenL3	63.76 ± 0.23	62.62 ± 0.22	64.65 ± 0.33	67.18 ± 0.43	67.69 ± 0.89	69.06 ± 0.42
eval set	PANN	56.18 ± 0.13	60.08 ± 0.08	57.83 ± 1.01	63.11 ± 0.48	61.63 ± 0.48	65.29 ± 0.39
eval set	Kumar	60.00 ± 0.12	60.56 ± 0.13	61.56 ± 0.27	61.27 ± 0.30	63.42 ± 0.48	64.05 ± 0.27

Table 24: Harmonic means of **AUC-ROCs** for different input representations. © 2023 IEEE

dataset	VGGish	OpenL3	PANN	Kumar	no embedding
dev set	65.78 ± 0.37	70.94 ± 1.36	66.30 ± 0.55	65.85 ± 0.83	81.36 ± 0.66
eval set	64.69 ± 0.34	69.06 ± 0.42	65.29 ± 0.39	64.05 ± 0.27	73.43 ± 0.54

compared. It can be seen that training a shallow classifier results in significantly better performance than when not doing so. As for the directly trained system, sub-cluster AdaCos in combination with using the cosine distance leads to the best results for all four pre-trained embedding types.

As a last experiment, the best performances obtained with pre-trained embeddings are compared to the previously presented **ASD** system that is directly trained on the input data. The experimental results can be found in [Table 24](#) and the following observations can be made: First of all, training a model directly on the data leads to much better results than using pre-trained embeddings. The most likely reason is that pre-trained embeddings do not capture subtle differences between normal and anomalous samples similar to embeddings obtained with one-class losses. This is also supported by the findings in [\[67\]](#), stating that very noisy recordings are a problem for models that have not been trained on the same dataset. Second, **OpenL3** embedding perform second best and **VGGish**, **PANN** and **Kumar** embeddings perform worse while having very similar results. A possible explanation may be that these three embeddings are trained using a supervised training objective and **OpenL3** embeddings are the only embeddings resulting from **SSL**. This observation has also been made in [\[67\]](#).

Overall, the results indicate that using pre-trained models does not help to improve the performance for **ASD** tasks as the underlying models are too general

even when only a few data samples are available. However, using a sufficient number of training samples for the source domain results in an embedding space that also yields a reasonable performance on the target domain. In [Section 5.5](#), more experiments with pre-trained embeddings will be conducted for few-shot [OSC](#).

5.6 ADAPROJ

In [Section 3.4](#), it was shown that the sub-cluster AdaCos loss, which allows the embedding model to learn less restrictive distributions by utilizing multiple centers for each class, improves the resulting [ASD](#) performance. The idea of the AdaProj loss, which will be presented in this section, is to further generalize the sub-cluster AdaCos loss by enlarging the space of optimal solutions. More concretely, the distance to linear subspaces spanned by the centers, which act as basis vectors, instead of the distance to the centers themselves is measured when training the embedding model. This extends the set of optimal solutions from a few points, namely the sub-clusters, to entire linear sub-spaces for each class and allows the network to learn more complex distributions of the normal data samples. As a result, it is expected that this helps to identify anomalies in the embedding space. Another advantage is that the embedding model has more freedom to solve multiple classification tasks at once imposed by different types of meta information, which may be related to very different characteristics of the input space and thus should also not manifest in the same geometric properties in the embedding space. For example, one would expect that recordings belonging to different machine types are more different to each other than recordings of the same machine with different parameter settings.

Other works also use orthogonal projections onto sub-spaces when learning embeddings for related applications. In [\[268\]](#), two different orthogonal sub-spaces for normal and anomalous data are explicitly enforced through a loss function when training an autoencoder for detecting anomalous images. For semi-supervised image classification, class-specific subspace projections are learned by using a reconstruction loss in combination with a discriminative loss to ensure that these sub-spaces are different [\[125\]](#). Still, both approaches are very different from AdaProj as autoencoders need to also reconstruct the noise and non-target sound events, which degrades the performance of an [ASD](#) system and thus should be avoided.

5.6.1 Definition

Before defining the AdaProj loss, additional notation will be introduced.

Definition 5.1 (Projection onto linear span). Let $e \in \mathbb{R}^D$ be an embedding and let $C \subset \mathbb{R}^D$ denote basis vectors. Then, the *projection* $P_{\text{span}(C)}$ onto the linear span

of the embedding space, denoted by $\text{span}(\mathbf{C}) \subset \mathbb{R}^D$, containing all finite linear combinations of the basis vectors is defined as

$$\begin{aligned} \mathbf{P}_{\text{span}(\mathbf{C})} : \mathbb{R}^D &\rightarrow \text{span}(\mathbf{C}) \\ \mathbf{P}_{\text{span}(\mathbf{C})}(\mathbf{e}) &:= \sum_{\mathbf{c} \in \mathbf{C}} \langle \mathbf{e}, \mathbf{c} \rangle \mathbf{c}. \end{aligned} \quad (54)$$

Using this notation, the formal definition of the AdaProj loss is as follows.

Definition 5.2 (AdaProj). Using the same notation as introduced in Definitions 3.2 and 3.4, define the logit for class $k \in \{1, \dots, N_{\text{classes}}\}$ measuring the distance between an embedding and its projection to a linear subspace as

$$\begin{aligned} \mathbf{d}_{\text{proj}} : \mathbb{R}^D \times \mathcal{P}(\mathbb{R}^D) &\rightarrow \mathbb{R}_+ \\ \mathbf{d}_{\text{proj}}(\phi(\mathbf{x}, \mathbf{w}), \mathbf{C}_k) &:= \hat{\mathbf{s}} \cdot \|\mathbf{P}_{\mathcal{S}^{D-1}}(\phi(\mathbf{x}, \mathbf{w})) - \mathbf{P}_{\mathcal{S}^{D-1}}(\mathbf{P}_{\text{span}(\mathbf{C}_k)}(\phi(\mathbf{x}, \mathbf{w})))\|_2^2. \end{aligned} \quad (55)$$

Then, the *AdaProj* loss is defined as

$$\begin{aligned} \mathcal{L}_{\text{proj}} : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{P}(\mathbb{R}^D)) \times \Phi \times \mathbf{W} \times \Lambda(N_{\text{classes}}) &\rightarrow \mathbb{R}_+ \\ \mathcal{L}_{\text{proj}}(\mathcal{Y}, \mathcal{C}, \phi, \mathbf{w}, \text{lab}) &:= -\frac{1}{|\mathcal{Y}|} \sum_{\mathbf{x} \in \mathcal{Y}} \sum_{j=1}^{N_{\text{classes}}} \text{lab}(\mathbf{x})_j \log(\text{softmax}(\hat{\mathbf{s}} \cdot \mathbf{d}_{\text{proj}}(\phi(\mathbf{x}, \mathbf{w}), \mathbf{C}_j))) \end{aligned} \quad (56)$$

where $\tilde{\mathbf{s}} \in \mathbb{R}_+$ is the dynamically adaptive scale parameter of the AdaCos loss. For the AdaProj loss, the centers \mathcal{C} are called *basis vectors*.

Remark. Since all embeddings and the linear subspaces are projected onto the unit sphere, Lemma 3.1 implies that

$$\|\mathbf{P}_{\mathcal{S}^{D-1}}(\mathbf{x}) - \mathbf{P}_{\mathcal{S}^{D-1}}(\mathbf{P}_{\text{span}(\mathbf{C}_k)}(\mathbf{x}))\|_2^2 = 2(1 - \text{sim}(\mathbf{P}_{\mathcal{S}^{D-1}}(\mathbf{x}), \mathbf{P}_{\mathcal{S}^{D-1}}(\mathbf{P}_{\text{span}(\mathbf{C}_k)}(\mathbf{x}))).$$

This is the reason why the AdaProj loss is an angular margin loss.

The following Lemma formally proves the claim that the space of optimal solutions for the AdaProj loss is indeed larger than the solution space of the sub-cluster AdaCos loss.

Lemma 5.3. *Let $\mathbf{e} \in \mathbb{R}^D$ and let $\mathbf{C} \subset \mathbb{R}^D$ contain pairwise orthonormal elements. If $\mathbf{e} \in \text{span}(\mathbf{C}) \cap \mathcal{S}^{D-1}$, then*

$$\|\mathbf{P}_{\mathcal{S}^{D-1}}(\mathbf{e}) - \mathbf{P}_{\mathcal{S}^{D-1}}(\mathbf{P}_{\text{span}(\mathbf{C})}(\mathbf{e}))\|_2^2 = 0. \quad (57)$$

Proof. Let $\mathbf{e} \in \text{span}(\mathbf{C}) \cap \mathcal{S}^{D-1} \subset \mathbb{R}^D$ with $|\mathbf{C}| = N_{\text{centers}}$. Therefore, $\|\mathbf{e}\|_2 = 1$ and there are $\lambda_j \in \mathbb{R}$ such that $\mathbf{e} = \sum_{j=1}^{N_{\text{centers}}} \lambda_j \mathbf{c}_j$. Hence, it holds that

$$\begin{aligned} \mathbf{e} &= \sum_{j=1}^{N_{\text{centers}}} \lambda_j \mathbf{c}_j = \sum_{j=1}^{N_{\text{centers}}} \sum_{i=1}^{N_{\text{centers}}} \lambda_i \langle \mathbf{c}_i, \mathbf{c}_j \rangle \mathbf{c}_j = \sum_{j=1}^{N_{\text{centers}}} \left\langle \sum_{i=1}^{N_{\text{centers}}} \lambda_i \mathbf{c}_i, \mathbf{c}_j \right\rangle \mathbf{c}_j \\ &= \sum_{j=1}^{N_{\text{centers}}} \langle \mathbf{e}, \mathbf{c}_j \rangle \mathbf{c}_j = \mathbf{P}_{\text{span}(\mathbf{C})}(\mathbf{e}), \end{aligned} \quad (58)$$

which finishes the proof. \square

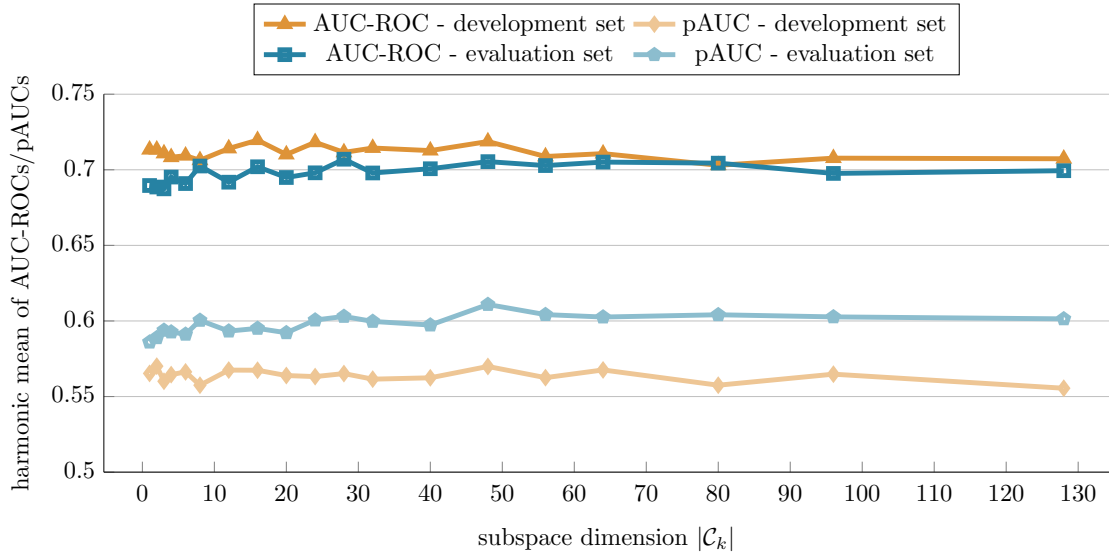


Figure 24: Domain-independent performance obtained on the DCASE2023 dataset with different subspace dimensions for the AdaProj loss. The means over ten independent trials are shown. © 2024 IEEE

For each class-specific subspace, the randomly sampled basis vectors are approximately orthogonal with very high probability [65]. Therefore, each subspace has a dimension of $|C|$ with very high probability and thus $|C| < D$ needs to be ensured to not allow the entire embedding space as a solution. Still, the size of the solution space for the AdaProj loss is much higher than for the sub-cluster AdaCos loss, for which only the sub-clusters themselves are optimal solutions.

5.6.2 Choosing a sub-space dimension

The impact of altering the subspace dimension of the AdaProj loss on the resulting performance is depicted in Figure 24. For all experimental evaluations in this and the next subsection, the ASD system presented in Section 5.3.2 with a few modifications is used. Most importantly, the sub-cluster AdaCos loss is replaced with AdaProj when training the embedding model. Additionally, the embedding dimension of the sub-networks is increased from 128 to 256 and 32 instead of 16 clusters are used when applying k-means to the embeddings belonging to the normal training samples of the source domain. The reason for increasing the dimensions of the sub-spaces is to grant the embedding model more freedom in learning class-specific distributions.

When inspecting the experimental results contained in Figure 24, it can be seen that for low to medium subspace dimensions the performance is approximately constant on the development set. On the evaluation set, medium subspace dimensions lead to slightly better performance than low subspace dimensions. For both dataset splits, the performance starts to degrade when being higher than approxi-

mately 48. In conclusion, the chosen subspace dimension should be neither too high nor too low. Therefore, a dimension of 32 was used in the following experiments.

5.6.3 Performance evaluation

As a next step, the AdaProj loss is compared to other loss functions on the DCASE2022 and the DCASE2023 dataset by only replacing the loss function of the embedding model and keeping all other components of the system the same. The results can be found in [Table 25](#). Overall, the AdaProj loss outperforms all other loss functions, especially on the evaluation split of the DCASE2023 dataset. A possible explanation is that the classification task on the DCASE2023 dataset is simpler than on the DCASE2022 dataset due to the reduced number of classes and thus the embedding model may be able to learn solutions that are almost trivial for certain classes. This means that for these classes the embeddings do not contain enough information to be able to discriminate between embeddings of normal and anomalous samples. Since the solution spaces of the AdaProj loss are much larger, a trivial solution only needs to be contained in a sub-space but the learned distribution may have a very complex structure inside this space, which allows to still be able to detect anomalies.

As a second observation, one can see that the sub-cluster AdaCos performs worse than AdaCos, which seems to contradict the results presented in [Section 3.4](#). However, in the experiments conducted in [Section 3.4](#) the centers have been adapted during training, which is not the case in these experiments. Without adapting the centers, all sub-clusters have approximately the same distance to each other because they are very likely to be orthogonal regardless of whether they belong to the same target class or not. This has two consequences: First, the sub-cluster AdaCos loss is actually more restrictive than the AdaCos loss and results in learning more compact distributions for each class to ensure low inter-class similarity to all sub-clusters of the other classes. This makes it difficult to distinguish between normal and anomalous samples. Second, in most cases only a single sub-cluster for each class is utilized by the model to ensure high intra-class similarity while still ensuring low inter-class similarity. This favors learning even more compact distributions for the normal samples.

5.7 SELF-SUPERVISED LEARNING

Throughout this thesis, it has been shown that the performance of an ASD system degrades when decreasing the amount of meta information used for training the embedding model. This difficulty is one focus of the task imposed by the DCASE2023 ASD dataset, which contains only a single section for each machine type. To still be able to learn informative embeddings yielding a good performance, the difficulty of the auxiliary classification task needs to be increased. Self-supervised learning (SSL) [132] is a type of unsupervised learning that does not require any class labels

Table 25: ASD performance obtained with different loss functions on the DCASE2022 and DCASE2023 datasets. Harmonic means of all AUC-ROCs and pAUCs over all sections of each dataset are depicted in percent. Arithmetic mean and standard deviation of the results over ten independent trials are shown. Best results in each column are highlighted with bold letters. © 2024 IEEE

DCASE2022 development set						
loss function	source domain		target domain		domain-independent	
	AUC-ROC	pAUC	AUC-ROC	pAUC	AUC-ROC	pAUC
intra-class compactness loss	81.8 ± 1.6	74.9 ± 1.7	75.3 ± 1.0	63.4 ± 0.6	79.2 ± 0.9	64.7 ± 1.1
intra-class compactness loss + CXE	82.5 ± 1.8	75.5 ± 0.9	75.5 ± 0.7	61.6 ± 0.9	79.0 ± 0.8	65.0 ± 0.7
AdaCos loss	82.6 ± 1.4	76.0 ± 1.1	76.5 ± 1.2	62.3 ± 1.4	79.8 ± 0.7	65.5 ± 0.9
sub-cluster AdaCos loss	83.2 ± 2.1	75.9 ± 1.3	77.6 ± 1.0	62.1 ± 1.5	80.0 ± 1.4	65.2 ± 1.1
AdaProj loss	84.3 ± 1.1	76.3 ± 1.1	77.2 ± 1.2	62.2 ± 1.1	80.6 ± 0.8	65.5 ± 1.3
DCASE2022 evaluation set						
loss function	source domain		target domain		domain-independent	
	AUC-ROC	pAUC	AUC-ROC	pAUC	AUC-ROC	pAUC
intra-class compactness loss	74.7 ± 0.9	64.2 ± 1.3	65.9 ± 0.8	57.8 ± 0.9	70.3 ± 0.8	58.9 ± 0.8
intra-class compactness loss + CXE	75.6 ± 0.7	66.9 ± 0.8	69.3 ± 0.7	59.3 ± 0.7	72.6 ± 0.4	60.3 ± 0.7
AdaCos loss	77.2 ± 0.5	65.9 ± 1.4	68.6 ± 1.1	58.6 ± 0.7	73.0 ± 0.4	59.7 ± 0.6
sub-cluster AdaCos loss	77.0 ± 0.7	66.5 ± 0.9	68.3 ± 0.8	58.8 ± 0.6	72.9 ± 0.6	59.5 ± 0.5
AdaProj loss	77.4 ± 1.0	67.0 ± 0.6	69.7 ± 0.6	59.6 ± 0.6	73.6 ± 0.7	60.5 ± 0.7
DCASE2023 development set						
loss function	source domain		target domain		domain-independent	
	AUC-ROC	pAUC	AUC-ROC	pAUC	AUC-ROC	pAUC
intra-class compactness loss	67.0 ± 2.1	62.4 ± 1.0	69.1 ± 1.4	56.4 ± 1.1	67.7 ± 1.2	56.9 ± 0.9
intra-class compactness loss + CXE	70.6 ± 1.8	64.1 ± 1.8	71.2 ± 1.4	55.5 ± 1.6	70.4 ± 1.0	57.4 ± 1.1
AdaCos loss	70.7 ± 1.3	64.3 ± 1.1	71.2 ± 1.1	55.4 ± 1.3	70.9 ± 0.9	56.8 ± 0.9
sub-cluster AdaCos loss	68.3 ± 1.7	62.0 ± 1.5	71.8 ± 1.5	55.6 ± 1.5	70.4 ± 0.9	56.3 ± 0.8
AdaProj loss	70.3 ± 1.7	61.8 ± 1.6	72.2 ± 1.4	55.1 ± 1.1	71.4 ± 1.0	56.2 ± 0.7
DCASE2023 evaluation set						
loss function	source domain		target domain		domain-independent	
	AUC-ROC	pAUC	AUC-ROC	pAUC	AUC-ROC	pAUC
intra-class compactness loss	73.5 ± 1.8	63.4 ± 1.8	58.8 ± 2.5	55.7 ± 1.3	64.0 ± 1.5	55.8 ± 0.9
intra-class compactness loss + CXE	74.3 ± 1.5	64.0 ± 1.6	61.6 ± 2.0	55.7 ± 0.9	67.5 ± 0.8	57.5 ± 1.0
AdaCos loss	74.7 ± 1.5	63.8 ± 1.8	61.6 ± 3.4	57.1 ± 1.4	68.0 ± 1.6	58.0 ± 1.1
sub-cluster AdaCos loss	73.2 ± 1.9	61.6 ± 1.4	62.0 ± 2.2	55.8 ± 1.3	66.5 ± 1.6	56.2 ± 1.0
AdaProj loss	74.2 ± 1.8	62.9 ± 1.0	64.4 ± 2.0	57.7 ± 0.8	69.8 ± 1.3	60.0 ± 0.5
arithmetic mean over all datasets						
loss function	source domain		target domain		domain-independent	
	AUC-ROC	pAUC	AUC-ROC	pAUC	AUC-ROC	pAUC
intra-class compactness loss	74.3	66.2	67.3	58.3	70.3	59.1
intra-class compactness loss + CXE	75.8	67.6	69.4	58.0	72.4	60.1
AdaCos loss	76.3	67.5	69.5	58.4	72.9	60.0
sub-cluster AdaCos loss	75.4	66.5	69.9	58.1	72.5	59.3
AdaProj loss	76.6	66.3	70.9	58.7	73.9	60.6

but augments the data in specific ways and trains the network to recognize these augmentations. To be able to correctly recognize specific augmentations, the embedding model needs to learn the structure of the original data resulting in more information being captured that may also be useful to detect anomalous data. Other applications of [SSL](#) are to learn representations for speech-related [152] or general-purpose audio tasks [165, 167].

As already stated in [Section 2.7](#), there are also several works applying [SSL](#) to [ASD](#) using different data augmentation techniques. Examples are detecting pitch-shifted and time-stretched recordings [85], recognizing linear combinations of similar target sounds [134] generated with mixup [272], mixing first- and second-order statistics of time-frequency representations [26] and pre-training an autoencoder using a modified version of variance-invariance-covariance regularization [12]. Some works on [ASD](#) denote auxiliary classification tasks based on meta information [43, 60] as [SSL](#). However, since [SSL](#) tasks do not require any manually annotated labels, training paradigms completely relying on meta information should be called supervised tasks instead.

As seen in [Section 5.5](#), [openL3](#) embeddings, which are the only embeddings based on [SSL](#), performed best of the investigated pre-trained embeddings. This can be seen as additional evidence that applying [SSL](#) when directly training an embedding model on the data may improve the performance. Investigating this is the goal of this section.

5.7.1 Approaches

In this section, different [SSL](#) approaches for [ASD](#) will be presented. The first approach, called statistics exchange ([StatEx](#)) [26], was proposed to simulate anomalous samples that are used for training a discriminative [ASD](#) system. The anomalies are simulated by exchanging first and second order statistics over the time or frequency axis of the time-frequency representations of two random training samples. The following definition formalizes this.

Definition 5.4 (Statistics exchange (original)). Let $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^{\mathbf{T} \times \mathbf{F}}$ with $\mathbf{T}, \mathbf{F} \in \mathbb{N}$ denote the time-frequency representations of two random training samples. Let μ_1, μ_2 denote the first-order statistics over the time or frequency dimension of \mathbf{x}_1 and \mathbf{x}_2 , respectively, and σ_1, σ_2 denote the corresponding second-order statistics. Then, the augmented sample $\mathbf{x}_{\text{new}}(\mathbf{x}_1, \mathbf{x}_2)$ and its categorical label $\text{lab}(\mathbf{x}_{\text{new}}(\mathbf{x}_1, \mathbf{x}_2))$ are given by

$$\begin{aligned} \mathbf{x}_{\text{new}}(\mathbf{x}_1, \mathbf{x}_2) &:= \frac{\mathbf{x}_1 - \mu_1}{\sigma_1} \sigma_2 + \mu_2 \in \mathbb{R}^{\mathbf{T} \times \mathbf{F}} \\ \text{lab}(\mathbf{x}_{\text{new}}(\mathbf{x}_1, \mathbf{x}_2)) &:= (\text{lab}(\mathbf{x}_1)_1 \cdot \text{lab}(\mathbf{x}_2), \dots, \text{lab}(\mathbf{x}_1)_{\mathbf{N}_{\text{classes}}} \cdot \text{lab}(\mathbf{x}_2)) \in [0, 1]^{\mathbf{N}_{\text{classes}}^2}. \end{aligned} \tag{59}$$

If the statistics over the time axis are used, this approach is called *frequency StatEx*. If the statistics over the frequency axis are used, it is called *temporal StatEx*.

Remark. For the original definition of [StatEx](#), frequency bands or temporal regions smaller than the selected maximum size imposed by the dimensions of the spectrogram are used. For the sake of simplicity, the entire signals are used in this work as the difference in performance is neglectable.

When using [StatEx](#), another anomaly is simulated for each normal training sample contained in a batch. Both versions are used for training the embedding model, essentially doubling the batch size. For the normal training samples, the original classes are used. For the simulated anomalies, the number of classes increases quadratically. Depending on the number of original classes, training an embedding model may therefore quickly become infeasible. Because of this, the following variant of [StatEx](#) is proposed.

Definition 5.5 (Statistics exchange (variant)). Using the same notation as in Definition 5.4 and the same definition of the augmented sample $\mathbf{x}_{\text{new}}(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^{T \times F}$, the categorical label of \mathbf{x}_{new} is set to

$$\text{lab}(\mathbf{x}_{\text{new}}(\mathbf{x}_1, \mathbf{x}_2)) = (\mathbf{0}, 0.5 \cdot \text{lab}(\mathbf{x}_1), 0.5 \cdot \text{lab}(\mathbf{x}_2)) \in [0, 1]^{3N_{\text{classes}}} \quad (60)$$

where $\mathbf{0} = (0, \dots, 0) \in [0, 1]^{N_{\text{classes}}}$.

This [StatEx](#) variant has the advantage that only three times as many classes are needed instead of increasing the number of classes quadratically. Another modification is that a probability of 50% is used to decide whether to apply [StatEx](#) or not during training instead of using all original samples and their augmented version. This ensures that the batch size does not increase when using [StatEx](#). When not applying [StatEx](#), i.e. using an original training sample $\mathbf{x}_{\text{new}}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1$ the categorical label is set to

$$\text{lab}(\mathbf{x}_{\text{new}}(\mathbf{x}_1, \mathbf{x}_2)) := (\text{lab}(\mathbf{x}_1), \mathbf{0}, \mathbf{0}) \in [0, 1]^{3N_{\text{classes}}}. \quad (61)$$

Applying [TMN](#) sets the temporal mean of each sample to zero and thus impedes using frequency [StatEx](#). Therefore, only temporal [StatEx](#) is applied when training the embedding model while using [TMN](#) to pre-process the input features.

As a second [SSL](#) approach, feature exchange ([FeatEx](#)), which is inspired by the training procedure of the [OpenL3](#) embeddings [6, 7, 32], will be presented. The reader is reminded that these embeddings are trained by comparing video frames to audio clips with a length of 1s and training the network to predict whether the embeddings extracted from a frame and a clip belong together or not (cf. [Section 2.5](#)). Since the [ASD](#) system only utilizes audio data, directly applying this approach is impossible. Instead, the embeddings belonging to both feature branches are used. Formally, [FeatEx](#) is defined as:

Definition 5.6 (Feature exchange). For embedding model $\phi = (\phi_{\text{STFT}}, \phi_{\text{DFT}}) \in \Phi$ with parameter settings $\mathbf{w} = (\mathbf{w}_{\text{STFT}}, \mathbf{w}_{\text{DFT}}) \in \mathcal{W}$ and two random training samples $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}_{\text{train}}$, let $\phi(\mathbf{x}_1, \mathbf{w}) = (\phi_{\text{STFT}}(\mathbf{x}_1, \mathbf{w}_{\text{STFT}}), \phi_{\text{DFT}}(\mathbf{x}_1, \mathbf{w}_{\text{DFT}})) \in \mathbb{R}^D$ and

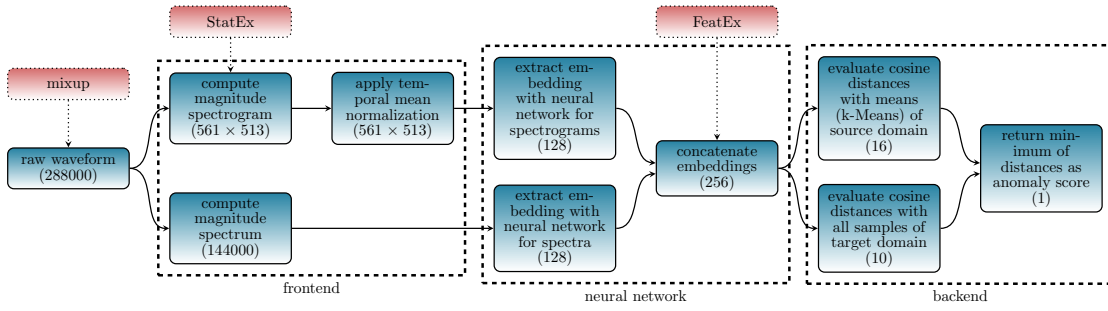


Figure 25: Illustration of an ASD system utilizing multiple SSL approaches. The baseline system is colored in blue and the SSL approaches are colored in red. Representation size in each step is given in brackets. © 2024 IEEE

$\phi(x_2, w) = (\phi_{\text{STFT}}(x_2, w_{\text{STFT}}), \phi_{\text{DFT}}(x_2, w_{\text{DFT}})) \in \mathbb{R}^D$ denote the concatenated embeddings of both sub-networks $\phi_{\text{STFT}}, \phi_{\text{DFT}}$ belonging to these samples. Then, an augmented embedding $e_{\text{new}}(x_1, x_2, \phi, w) \in \mathbb{R}^D$ and its categorical label are set to

$$\begin{aligned} e_{\text{new}}(x_1, x_2, \phi, w) &= (\phi_{\text{STFT}}(x_1, w_{\text{STFT}}), \phi_{\text{DFT}}(x_2, w_{\text{DFT}})) \in \mathbb{R}^D \\ \text{lab}(e_{\text{new}}(x_1, x_2, \phi, w)) &= (\mathbf{0}, 0.5 \cdot \text{lab}(x_1), 0.5 \cdot \text{lab}(x_2)) \in [0, 1]^{3N_{\text{classes}}} \end{aligned} \quad (62)$$

where $\mathbf{0} = (0, \dots, 0) \in [0, 1]^{N_{\text{classes}}}$.

As for the StatEx variant, the number of classes is tripled when applying FeatEx and a probability of 50% is used to decide whether to apply FeatEx or not.

5.7.2 Combining multiple approaches

Each SSL approach forces the embedding model to capture more information in the embeddings by increasing the difficulty of the classification task the model needs to solve. Therefore, multiple SSL approaches can all be used for training a single system. The system presented in Section 5.3.1 using the StatEx variant, FeatEx as well as mixup [272] (see Section 2.7.3) is illustrated in Figure 25. For the experimental evaluations in the following section, mixup is only used with a probability of 50% when also using any other SSL approach. If the model is trained without StatEx or FeatEx, mixup is applied with a probability of 100% to each training sample. The model is trained with a combined loss function consisting of two classification losses. The first loss is the standard classification task based on provided meta information and uses the mixed-up labels for training. The second loss is based on the sequentially applied SSL approaches and thus uses $9 \cdot N_{\text{classes}}$ total classes. Both losses are realized with sub-cluster AdaCos losses with 16 sub-clusters for each class. As before, non-trainable class centers are used for the standard classification task. For the SSL loss, trainable class centers are used as the embedding model should learn to detect augmented samples as pseudo-

anomalies close to the distributions of the normal, non-augmented samples to refine the boundary of these distributions.

Note that, although mixup utilizes class labels, it also imposes an additional self-supervised task of needing to estimate the mixing coefficient of linear interpolations between random training samples. Hence, it can be seen as a label-dependent [SSL](#) approach. In theory, mixup could also be applied without using class labels and teaching the model to only estimate the mixing coefficient. But as previously shown, utilizing meta information is important to effectively handle the noise and thus a classification task should be used.

5.7.3 *Performance evaluation*

As a first experiment, the performances obtained with individual [SSL](#) approaches and the combined approach are compared to the baseline performance obtained when not using any [SSL](#). The results can be found in [Table 26](#). Overall, it can be seen that the [SSL](#) approaches significantly improve the performance of the baseline system. Furthermore, one can observe that, in general, [FeatEx](#) performs better than [StatEx](#). When only using [StatEx](#) without the standard classification task, the performance decreases as opposed to only using [FeatEx](#), which slightly improves the performance. This indicates that the proposed [FeatEx](#) approach is superior to [StatEx](#). When combining any [SSL](#) approach with the regular classification loss, both approaches, [FeatEx](#) and [StatEx](#), perform better compared to only using the baseline system, which shows that applying [SSL](#) is highly beneficial. Moreover, combining both [SSL](#) approaches seems to marginally improve performance over only applying one of them. Another observation is that the performance improvements are much bigger on the DCASE2023 dataset than on the DCASE2022 dataset. This is consistent with the experimental findings for the AdaProj loss. Once more, the most likely reason is that a less difficult classification task caused by less available meta information leads to less informative embeddings. Therefore, essential information needed to discriminate between embeddings of normal and anomalous data is not captured, which degrades performance.

In a second experiment, whose results can be found in [Table 27](#), three design choices are investigated: 1) using class labels based on the meta information for the [SSL](#) losses, 2) using non-trainable centers for the [SSL](#) losses and 3) not applying [TMN](#) but also using temporal [StatEx](#). Comparing the performances obtained when altering any of these design choices to the performance obtained with the original system, one can see that all changes lead to worse performance, especially on the evaluation set. This justifies the proposed design of the [ASD](#) system and the [SSL](#) approaches.

Table 26: Harmonic means of **AUC-ROCs** and **pAUCs** taken over all machine IDs obtained when using different **SSL** approaches. Highest **AUC-ROCs** and **pAUCs** in each column are highlighted in bold letters. Arithmetic mean and standard deviation over five independent trials are shown. © 2024 IEEE

DCASE2022 development set						
SSL approach	source domain		target domain		domain-independent	
	AUC-ROC	pAUC	AUC-ROC	pAUC	AUC-ROC	pAUC
baseline	84.2 ± 0.8%	76.5 ± 0.9%	78.5 ± 0.9%	62.5 ± 0.9%	81.4 ± 0.7%	66.6 ± 0.9%
StatEx variant	80.5 ± 1.8%	69.5 ± 1.9%	75.3 ± 1.7%	60.0 ± 0.9%	76.4 ± 1.5%	62.4 ± 1.2%
FeatEx	82.1 ± 0.9%	72.8 ± 0.8%	77.2 ± 1.0%	62.8 ± 0.6%	78.5 ± 0.6%	65.2 ± 0.6%
regular and StatEx variant	85.2 ± 0.9%	77.5 ± 1.2%	78.9 ± 0.9%	63.2 ± 1.6%	82.2 ± 0.6%	67.0 ± 1.0%
regular and FeatEx	85.1 ± 0.9%	76.3 ± 1.8%	77.9 ± 0.9%	62.7 ± 0.8%	81.6 ± 0.7%	67.0 ± 0.9%
combined approach	86.0 ± 0.9%	77.6 ± 0.8%	78.2 ± 0.7%	64.4 ± 1.1%	82.5 ± 0.8%	68.2 ± 1.1%
DCASE2022 evaluation set						
SSL approach	source domain		target domain		domain-independent	
	AUC-ROC	pAUC	AUC-ROC	pAUC	AUC-ROC	pAUC
baseline	76.8 ± 0.8%	65.8 ± 0.2%	69.8 ± 0.5%	59.7 ± 1.1%	73.4 ± 0.5%	59.8 ± 0.8%
StatEx variant	74.2 ± 0.6%	61.6 ± 1.4%	70.6 ± 0.5%	59.0 ± 0.7%	72.2 ± 0.3%	58.2 ± 0.7%
FeatEx	76.3 ± 0.9%	64.5 ± 1.2%	72.3 ± 0.6%	61.0 ± 0.7%	73.9 ± 0.5%	60.0 ± 0.9%
regular and StatEx variant	76.9 ± 0.4%	65.8 ± 0.9%	71.2 ± 0.3%	60.3 ± 0.7%	73.9 ± 0.3%	59.9 ± 0.6%
regular and FeatEx	78.1 ± 0.4%	67.0 ± 1.1%	72.2 ± 0.4%	61.3 ± 0.5%	74.9 ± 0.4%	61.5 ± 0.6%
combined approach	77.7 ± 0.8%	67.0 ± 0.5%	71.6 ± 1.0%	61.2 ± 0.9%	74.2 ± 0.3%	61.2 ± 0.3%
DCASE2023 development set						
SSL approach	source domain		target domain		domain-independent	
	AUC-ROC	pAUC	AUC-ROC	pAUC	AUC-ROC	pAUC
baseline	69.8 ± 1.8%	60.9 ± 0.9%	72.3 ± 1.8%	55.6 ± 0.9%	71.3 ± 0.6%	56.1 ± 0.8%
StatEx variant	67.8 ± 1.5%	59.2 ± 0.8%	69.7 ± 1.7%	54.7 ± 1.1%	69.0 ± 1.2%	55.7 ± 1.0%
FeatEx	68.4 ± 1.0%	60.2 ± 0.5%	74.4 ± 0.7%	57.6 ± 1.0%	71.7 ± 0.4%	57.5 ± 0.7%
regular and StatEx variant	70.3 ± 1.8%	62.0 ± 1.6%	72.2 ± 1.4%	56.2 ± 1.2%	71.2 ± 0.7%	57.0 ± 1.4%
regular and FeatEx	72.9 ± 2.0%	63.0 ± 1.3%	75.7 ± 0.8%	57.0 ± 1.6%	74.4 ± 1.0%	58.0 ± 1.4%
combined approach	71.2 ± 1.6%	62.7 ± 1.3%	75.0 ± 1.5%	56.1 ± 1.4%	73.1 ± 0.9%	57.3 ± 0.6%
DCASE2023 evaluation set						
SSL approach	source domain		target domain		domain-independent	
	AUC-ROC	pAUC	AUC-ROC	pAUC	AUC-ROC	pAUC
baseline	72.5 ± 0.8%	62.4 ± 1.2%	63.1 ± 2.6%	57.5 ± 0.8%	67.9 ± 1.0%	58.8 ± 0.8%
StatEx variant	70.0 ± 1.2%	59.7 ± 0.9%	66.7 ± 1.8%	58.4 ± 0.7%	65.8 ± 0.6%	57.1 ± 0.8%
FeatEx	69.3 ± 2.0%	59.3 ± 1.2%	69.1 ± 1.3%	59.0 ± 1.3%	68.1 ± 1.2%	58.1 ± 0.9%
regular and StatEx variant	72.4 ± 2.4%	62.4 ± 1.3%	65.9 ± 1.9%	59.0 ± 1.4%	69.5 ± 1.8%	60.7 ± 0.9%
regular and FeatEx	75.9 ± 1.0%	62.9 ± 1.3%	66.5 ± 1.9%	58.3 ± 1.0%	71.1 ± 1.1%	60.1 ± 1.3%
combined approach	75.5 ± 0.8%	64.5 ± 0.6%	68.7 ± 2.2%	59.3 ± 0.7%	72.6 ± 0.7%	61.6 ± 0.5%

Table 27: Harmonic means of **AUC-ROCs** and **pAUCs** taken over all machine types obtained on the DCASE2023 dataset by modifying design choices of the proposed **SSL**-based system. Arithmetic mean and standard deviation over five independent trials are shown. © 2024 IEEE

split	domain	SSL loss without class labels		non-trainable class centers		no TMN and full StatEx	
		AUC-ROC	pAUC	AUC-ROC	pAUC	AUC-ROC	pAUC
dev	source	70.8 ± 1.5%	63.2 ± 1.1%	71.5 ± 0.9%	64.8 ± 1.9%	70.9 ± 0.7%	61.0 ± 1.5%
dev	target	74.7 ± 1.5%	58.1 ± 1.6%	74.0 ± 2.0%	56.7 ± 1.0%	72.1 ± 1.4%	55.2 ± 1.0%
dev	mixed	72.3 ± 1.2%	57.9 ± 1.3%	71.6 ± 1.1%	57.7 ± 0.7%	71.3 ± 0.7%	55.6 ± 0.9%
eval	source	73.5 ± 2.4%	63.8 ± 0.6%	74.2 ± 0.7%	63.9 ± 1.3%	73.8 ± 1.3%	62.4 ± 1.5%
eval	target	62.1 ± 1.5%	57.7 ± 0.9%	58.2 ± 3.3%	57.3 ± 0.9%	66.9 ± 2.4%	58.5 ± 1.9%
eval	mixed	68.6 ± 1.2%	59.1 ± 0.7%	65.0 ± 0.9%	57.7 ± 0.6%	70.9 ± 0.8%	59.9 ± 0.8%

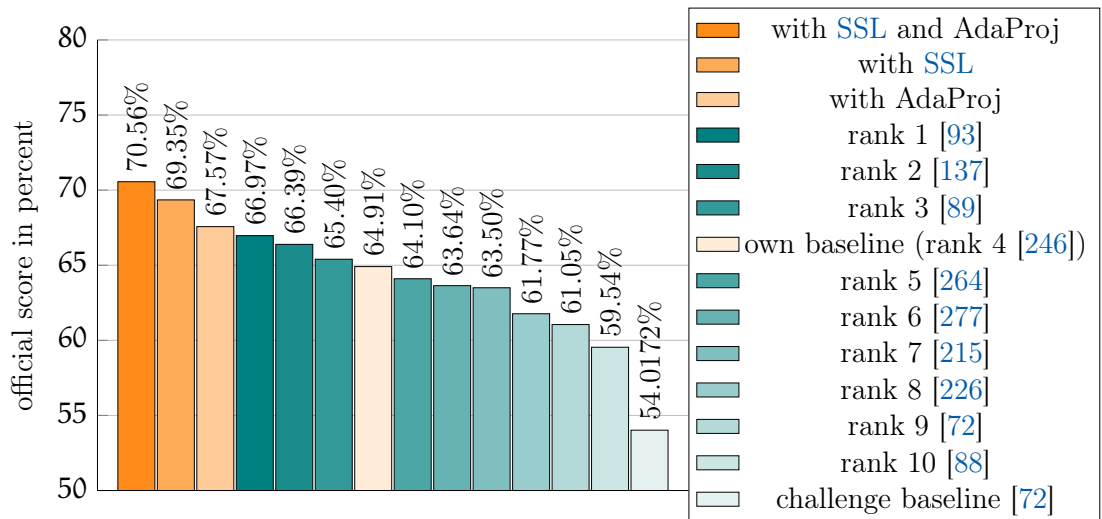


Figure 26: Comparison between the performances of the presented systems and official scores of the ten top-performing systems of the DCASE2023 Challenge. For the evaluations, ensembles consisting of ten sub-systems were used.

5.8 PUTTING IT ALL TOGETHER

The goal of this section is to compare the performance obtained with the presented systems of this chapter to the state-of-the-art performance. Furthermore, the AdaProj loss and SSL are combined in a single system by replacing both sub-cluster AdaCos losses of the system used in [Section 5.7](#) with AdaProj losses to see if this improves performance. For all evaluated systems, ensembles are used by taking the sum of the anomaly scores belonging to 10 individually trained versions of the same system. A comparison between the systems presented in this chapter and the ten top-performing systems of the DCASE2023 Challenge can be found in [Figure 26](#). It can be seen that the baseline system presented in [Section 5.3.1](#)

reached rank 4 and thus yields good but the best performance. When only replacing the loss functions with the AdaProj loss, the system slightly outperforms all other systems and thus yields state-of-the-art performance. However, the performance improvements obtained when applying the SSL approaches are significantly greater and the resulting system outperforms all other published systems by a large margin. Moreover, when combining AdaProj and SSL into a single system the resulting performance is even better. In conclusion, all approaches presented in this chapter are highly beneficial to improve the performance of a semi-supervised ASD system in domain-shifted conditions.

5.9 SUMMARY

In this chapter, the ASD system presented in Chapter 3 was modified to be robust to domain shifts from a source to a target domain, which are any changes of the distribution of normal data caused by altering the target sounds or the background noise. It was shown that many design choices of the system have an impact on the resulting performance. More concretely, it was shown that 1) preventing trivial solutions by not using bias terms or trainable centers, 2) utilizing STFT- and DFT-based input feature representations in combination with TMN with two sub-networks, whose embeddings are combined by concatenating them, as well as 3) using the cosine similarity as a backend all significantly improve the performance in domain-shifted conditions.

Further investigations were aimed at explaining the results obtained with the presented ASD system. To this end, the impact of specific regions of the input features on the ASD performance using RISE and visualizing the resulting embedding spaces using t-SNE were investigated. Here, it could be seen that the embeddings obtained when using an auxiliary classification task capture more meaningful structures and thus normal embeddings can be much better separated from anomalous ones. This verifies the findings presented in Chapter 3. In contrast to the source domain, there are only very few training samples provided for the target domain in domain-shifted conditions. Although pre-trained embeddings are a promising approach in settings with limited training data, the experiments conducted in this chapter showed that directly training the embedding model leads to significantly better performance than using pre-trained embeddings as input to a shallow classifier.

In case only little meta information is available for training an embedding model with an auxiliary classification task, the ASD performance decreases. Two additional modifications of the ASD system were presented to improve the performance in such a setting. First, the angular margin loss AdaProj for learning class-specific sub-spaces was proposed. This loss is a generalization of the sub-cluster AdaCos loss and enables the model to learn class-specific distributions that are more complex than a combination of Gaussians. Using AdaProj increased the ability of the system to distinguish between embeddings belonging to normal and anomalous

samples. Second, multiple [SSL](#) approaches were used to increase the difficulty of the classification task and thus force the embedding model to capture more information. To this end, [FeatEx](#), which randomly exchanges the embeddings of both sub-networks between different training samples, was presented. It was shown that combining [FeatEx](#) with [StatEx](#), which exchanges first- and second-order statistics of two random training samples, significantly improved performance when training the embedding model with an auxiliary classification task. Utilizing [AdaProj](#) and [SSL](#) together, led to an even better performance outperforming all other published systems with a significant margin. As a result, the system presented in this chapter reached a new state-of-the-art performance on the DCASE2023 dataset.

The goal of this chapter is to investigate open-set classification (**OSC**) as another application of **ASD** than acoustic machine condition monitoring, which has been extensively studied in the previous chapters. As mentioned in the introduction of this thesis, **OSC** can be separated into two sub-tasks: closed-set classification (**CSC**) and anomaly detection (**AD**) [238]. In the previous chapters, it has been shown that angular margin losses are an excellent choice for detecting anomalous sounds. Furthermore, angular margin losses are specifically designed for **CSC** and thus angular margin losses are expected to also be an excellent choice to train models for **OSC**. Because of this, the difficulty of the **OSC** tasks investigated in this chapter will be increased by only considering settings in which only very few training samples are provided. For many applications, this is a more realistic scenario than having access to many training samples for each class because it substantially simplifies the data collection and labeling process. Furthermore, keyword spotting will be used as an application for sound event detection (**SED**), which demands to not only recognize the correct keyword class but also to determine the precise on- and offset of each sound event.

This chapter is structured as follows. First, the **ASD** system presented in [Section 5.3.1](#) will be evaluated for few-shot **OSC**. Similar to the experiments conducted in [Section 5.5](#), the resulting performance will be compared to performances obtained with pre-trained embeddings. Second, the loss function **TACos**, which is a modification of the AdaCos loss capable of learning embeddings with a temporal structure, will be presented and evaluated for few-shot keyword spotting.

6.1 CONTRIBUTIONS OF THE AUTHOR

The sections of this chapter are largely based on the following key publications:

- Kevin Wilkinghoff and Fabian Fritz. “On Using Pre-Trained Embeddings for Detecting Anomalous Sounds with Limited Training Data.” In: *31st European Signal Processing Conference*. IEEE, 2023, pp. 186–190. DOI: [10.23919/EUSIPCO58844.2023.10290003](https://doi.org/10.23919/EUSIPCO58844.2023.10290003).
- Kevin Wilkinghoff, Alessia Cornaggia-Urrigshardt, and Fahrettin Gökgöz. “Two-Dimensional Embeddings for Low-Resource Keyword Spotting Based on Dynamic Time Warping.” In: *14th ITG Conference on Speech Communication*. VDE-Verlag, 2021, pp. 9–13.
- Kevin Wilkinghoff and Alessia Cornaggia-Urrigshardt. “TACos: Learning Temporally Structured Embeddings for Few-Shot Keyword Spotting with

Dynamic Time Warping.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 9941–9945. DOI: [10.1109/ICASSP48485.2024.10445814](https://doi.org/10.1109/ICASSP48485.2024.10445814).

For publications that are not single-authored, individual contributions of the thesis author and all co-authors to these publications are stated in [Section A.1](#). If not stated otherwise, the content listed in the following paragraph is the sole contribution of the thesis author.

[Section 6.2](#) is based on [\[258\]](#). Fabian Fritz helped with the experimental evaluations by writing wrapper functions for the pre-trained embeddings. [Section 6.3](#) is mostly based on [\[254\]](#), which is an extension of [\[256\]](#). Thus, [\[256\]](#) is only indirectly covered. Alessia Cornaggia-Urrigshardt helped with creating the dataset presented in [Section 6.3.2](#) and applied [DTW](#) to the embeddings. Additionally, she evaluated the [HFCC](#)-based model presented in [Section 6.3.7](#) and conducted the experiments related to the global and individual decision thresholds contained in [Table 31](#).

6.2 FEW-SHOT OPEN-SET CLASSIFICATION

Before conducting any experiments, the terms *open-set classification* and *few-shot learning* will be formalized. [OSC](#) tasks can be characterized by the so-called *openness* [\[200\]](#), which quantifies *how open* a specific [OSC](#) task is by measuring the relation between known and unknown classes and is defined as follows:

Definition 6.1 (Openness). Let $N_{\text{classes}}^{\text{train}}, N_{\text{classes}}^{\text{test}} \in \mathbb{N}$ denote the number of classes of the training subset and the test subset, respectively. Then the *openness* of an [OSC](#) task is defined as

$$\text{openness}(N_{\text{classes}}^{\text{train}}, N_{\text{classes}}^{\text{test}}) := 1 - \sqrt{\frac{2 \cdot N_{\text{classes}}^{\text{train}}}{N_{\text{classes}}^{\text{train}} + N_{\text{classes}}^{\text{test}}}} \in [0, 1]. \quad (63)$$

Remark. $N_{\text{classes}}^{\text{train}}$ denotes the number of known target classes and known non-target classes (known unknowns), for which training samples are provided. $N_{\text{classes}}^{\text{test}}$ not only contains the same known target and non-target classes but may also contain additional non-target classes not encountered during training (unknown unknowns). These unknown unknowns correspond to the anomalies of a semi-supervised anomaly detection task and are the essential difference between closed-set and open-set problems. In case no unknown unknowns exist, the [OSC](#) task is in fact a [CSC](#) task because all classes are known during training. Note that for realistic applications it is usually impossible to provide training samples that fully capture the space of unknown classes as it is also the case in a supervised anomaly detection setting. For the same reason, even counting $N_{\text{classes}}^{\text{test}}$ for a realistic application is impossible. Because of this, $N_{\text{classes}}^{\text{test}}$ denotes the number of classes contained in a specific set of test samples, which is always less or equal than the real number of classes of the target application.

Since the openness is such an essential characteristic of an OSC task, it can be used to describe a task or serve as a metric to compare similar tasks. A higher openness indicates more unknown classes relative to the number of known classes. An openness of 0 corresponds to a CSC task and an openness of 1 corresponds to a finite number of known classes and an infinite number of unknown non-target classes, since $N_{\text{classes}}^{\text{train}} \neq 0$.

Few-shot learning [227] describes a task for which only $K \in \mathbb{N}$ training samples are available for each class (*K-shot learning*). Hence, the size of the training dataset is very limited, making it difficult to train a model such that it generalizes well to unseen data. The most popular loss function for few-shot classification [227] is a prototypical loss [209] defined as follows:

Definition 6.2 (Prototypical loss). Let $\mathbf{d} : X \times X \rightarrow \mathbb{R}_+$ denote a metric and $Y \subset X$ be finite. For training step $t \in \mathbb{N}_0$, let $\mathcal{S}^{(t)}, \mathcal{Q}^{(t)} \subset Y$ with $\mathcal{S}^{(t)} \cap \mathcal{Q}^{(t)} = \emptyset$ denote randomly sampled subsets called *support set* and *query set*, respectively. Furthermore, define the *prototypes* $\mathbf{C}^{(t)} = \{\mathbf{c}_1^{(t)}, \dots, \mathbf{c}_{N_{\text{classes}}}^{(t)}\} \subset \mathbb{R}^D$ as the means of the embeddings computed from all $\mathbf{x} \in \mathcal{S}^{(t)}$ belonging to the same class $i \in \{1, \dots, N_{\text{classes}}\}$, i.e. for embedding model $\phi \in \Phi$ with parameters $\mathbf{w} \in W$ set

$$\mathbf{c}_i^{(t)}(\phi, \mathbf{w}, \mathcal{S}^{(t)}) := \text{mean}(\{\phi(\mathbf{x}, \mathbf{w}) \in \mathbb{R}^D : \mathbf{x} \in \mathcal{S}^{(t)}, \text{class}(\mathbf{x}) = i\}). \quad (64)$$

Then, using the same notation as used in Definition 2.7, the *prototypical loss* at training step t is defined as

$$\begin{aligned} \mathcal{L}_{\text{prot}}^{(t)} : \mathcal{P}(X) \times \mathcal{P}(\mathbb{R}^D) \times \Phi \times W \times \Lambda(N_{\text{classes}}) &\rightarrow \mathbb{R}_+ \\ \mathcal{L}_{\text{prot}}^{(t)}(\mathcal{Q}^{(t)}, \mathbf{C}^{(t)}, \phi, \mathbf{w}, \text{lab}, \mathbf{d}) & \\ := - \frac{1}{|\mathcal{Q}^{(t)}|} \sum_{\mathbf{x} \in \mathcal{Q}^{(t)}} \sum_{j=1}^{N_{\text{classes}}} \text{lab}(\mathbf{x})_j \log(\text{softmax}(\mathbf{d}(\phi(\mathbf{x}, \mathbf{w}), \mathbf{c}_j^{(t)}))) & \end{aligned} \quad (65)$$

When choosing the Euclidean distance as a metric for the prototypical loss, i.e.

$$\mathbf{d}(\phi(\mathbf{x}, \mathbf{w}), \mathbf{c}_j^{(t)}) := \|\phi(\mathbf{x}, \mathbf{w}) - \mathbf{c}_j^{(t)}\|_2^2, \quad (66)$$

Theorem 3.5 and Corollary 3.3 show that angular margin losses such as the AdaCos loss can also be viewed as prototypical losses or angular prototypical losses [30] with a margin between classes. The main difference between these types of loss functions is that for a prototypical loss the centers are re-calculated during training by randomly sampling a support set whereas, for an angular margin loss, they are treated as trainable parameters or not adapted at all. The similarity of the loss functions is the reason why using an angular margin loss in a *few-shot* open-set classification setting is a viable option. However, as the number of training samples is very limited, learning more complex distributions than Gaussians for each class is not feasible and thus extensions of standard angular margin losses such as the

presented sub-cluster AdaCos loss (see [Section 3.4](#)) or AdaProj (see [Section 5.6](#)) should not be used. In addition, the [CSC](#) classification sub-task is challenging by itself and thus for [OSC](#) avoiding trivial solutions is less of a problem than for [ASD](#) alone.

In theory, the very limited number of training samples available for few-shot classification should favor pre-trained embeddings over training a system directly on the dataset. This motivates the goal of this section, which is to evaluate the previously proposed [ASD](#) system for few-shot [OSC](#) and compare its performance to performances obtained with pre-trained embeddings.

6.2.1 Dataset

The few-shot [OSC](#) dataset used in this section aims at detecting and correctly classifying acoustic alarms in domestic environments [162]. In total, the dataset contains 34 classes with 24 known alarm sounds and 10 unknown sounds belonging to one of the classes “car horn”, “clapping”, “cough”, “door slam”, “engine”, “keyboard tapping”, “music”, “pots and pans”, “steps” and “water falling”. For each class, there are 40 recordings, each with a length of 4 s and a sampling rate of 16 kHz. There are several different versions of this dataset defined by the number of shots available for each class (1, 2 or 4) and the openness (0, 0.04 or 0.09). Each version of the dataset is divided into a training and a test subset by using cross-validation. An overview can be found in [Table 28](#). The performance of a trained system is evaluated by using the weighted accuracy (cf. [Section 2.10](#), [Equation 27](#)) with $\beta_{\text{ACC}} = 0.5$ as a performance measure.

6.2.2 System design

To evaluate the performance on this few-shot [OSC](#) dataset, the [ASD](#) systems using pre-trained embeddings or the directly trained system presented in [Section 5.5](#) were slightly modified. First of all, only a single sub-cluster per class was used for the sub-cluster AdaCos loss because only very few training samples are available for each class. Depending on the specific version of the dataset, some hyperparameters of the system were adapted to obtain more robust decision thresholds, which were needed to compute the performance of an [ASD](#) system with a threshold-dependent evaluation metric. More concretely, the number of training epochs was set to 100 times the number of shots and the batch size was set to 8 times the number of shots. Furthermore, the decision threshold was set to 0.6, 0.65 or 0.75 for low, medium or high openness settings, respectively. As a last modification, the openL3 embeddings pre-trained on the music subset instead of the environmental subset were used because acoustic alarms are more similar to music than environmental sound events.

Table 28: Structure of the few-shot OSC dataset for acoustic alarm detection in domestic environments.

openness	shots	validation folds	number of classes		
			known	known unknown	unknown
low (0)	1	40	24	10	0
low (0)	2	20	24	10	0
low (0)	4	10	24	10	0
middle (0.04)	1	40	24	5	5
middle (0.04)	2	20	24	5	5
middle (0.04)	4	10	24	5	5
high (0.09)	1	40	24	0	10
high (0.09)	2	20	24	0	10
high (0.09)	4	10	24	0	10

6.2.3 Experimental results

A comparison of the performances obtained with different systems and different pre-trained embeddings can be found in Table 29. As expected, using more shots or having a lower openness leads to a better mean performance and a smaller variance as more training data should improve the performance and CSC is an easier problem than OSC. Furthermore, all proposed systems significantly outperform the two baseline systems proposed in [162] showing once more that the systems presented in this thesis are well designed. However, there is a strong difference in how well the systems perform. Overall, VGGish performs worst, followed by PANN, OpenL3 and Kumar embeddings and the directly trained system performs best. Moreover, all of the proposed systems have different strengths and weaknesses. The system based on OpenL3 embeddings performs best in the low openness setting, which is in fact a CSC task. In general, the directly trained model performs best for middle or high openness settings. An exception is the high openness setting when only a single shot is available for each class where Kumar embeddings perform best. The relative degradation of the performance caused by decreasing the number of shots is smaller when using pre-trained embeddings than with a directly trained model. This is expected since most parameters of the pre-trained systems belong to the embedding models, which are trained using external data, and thus, in total, only a few parameters need to be trained with the shots belonging to the application-dependent OSC dataset.

Table 29: Weighted accuracies (in percent) obtained with different systems and input representations for various openness settings and number of shots per class. Mean and standard deviation of five independent trials are shown. © 2023 IEEE

openness	shots	baselines [162]		proposed system using different input representations				
		OpenL3	YAMNet	VGGish	OpenL3	PANN	Kumar	directly trained
low	1	56.8	80.1	90.0 ± 2.2	98.1 ± 1.0	95.6 ± 1.4	96.5 ± 1.3	97.4 ± 1.1
low	2	90.3	88.2	95.6 ± 1.6	99.6 ± 0.3	97.7 ± 0.8	98.5 ± 0.9	99.1 ± 0.7
low	4	97.2	94.9	98.4 ± 0.7	99.9 ± 0.1	98.9 ± 0.5	99.6 ± 0.4	99.7 ± 0.4
middle	1	74.1	78.3	88.7 ± 2.1	97.0 ± 2.3	94.9 ± 1.7	96.1 ± 1.6	96.8 ± 1.4
middle	2	86.7	85.6	93.4 ± 1.8	99.2 ± 0.6	95.7 ± 1.9	97.8 ± 1.3	98.7 ± 0.8
middle	4	91.3	91.9	96.2 ± 1.6	99.3 ± 0.5	97.8 ± 1.1	98.6 ± 1.3	99.8 ± 0.2
high	1	49.9	57.1	84.0 ± 2.6	88.8 ± 5.3	92.1 ± 2.6	94.8 ± 2.4	92.6 ± 4.5
high	2	58.3	61.1	87.8 ± 2.6	94.0 ± 3.2	92.9 ± 2.9	97.0 ± 1.7	97.5 ± 1.7
high	4	60.5	64.3	87.8 ± 2.5	96.1 ± 1.5	96.0 ± 2.1	98.4 ± 1.3	99.1 ± 1.1
arithmetic mean		73.9	77.9	91.3	96.9	95.7	97.5	97.9

Once more, these results show that using pre-trained embeddings for detecting anomalies is not an optimal choice. Although the gap in performance is smaller for this few-shot OSC task than for the pure ASD (cf. Section 5.5) task and the dataset used here does not consist of noisy audio recordings, directly training a system leads to better performance. For CSC tasks, the situation is different and pre-trained embeddings such as OpenL3 embeddings may slightly outperform a directly trained system. However, CSC tasks are not the focus of this thesis.

6.3 SOUND EVENT DETECTION APPLICATION: KEYWORD SPOTTING

Keyword spotting (KWS) is the task of detecting all occurrences of a small set of pre-defined words, so-called *keywords*, with precise on- and offsets in audio signals of possibly long duration [136]. Applications are activating voice assistants [150, 199], querying large databases [157] or monitoring audio streams such as radio communication transmissions [147]. All KWS tasks are inherently OSC problems since each keyword defines another class and the keywords are only a small subset of the entire search space, which may contain arbitrary speech, silence or completely different sounds not related to speech. As strongly labeled training data, i.e. samples with annotated on- and offsets of the keywords is difficult and thus costly to obtain, often only a few training samples can be provided making it a few-shot task. The main difference to the previously presented few-shot open-set classification task is the need for detecting on- and offsets during inference, which adds another layer of difficulty.

As it is the case for [ASD](#) systems, state-of-the-art [KWS](#) systems are based on learning discriminative embeddings [94, 138]. For few-shot [KWS](#), these embeddings are also learned using a prototypical loss [98, 142, 175]. The same is true when using pre-trained embeddings [103] or for similar applications such as bio-acoustic event detection [169] or sound event detection in general [229, 230]. However, all of these networks require to choose a fixed input size resulting in the following dilemma: If the input size is too small, then insufficient information may be contained in the input data to classify correctly and the same instance may be detected multiple times. If the input size is too large, then too much irrelevant or contradicting information may be contained, which degrades the performance. Furthermore, multiple instances may be detected only once and precisely detecting on- and offsets of events is difficult. In practice, finding a balanced size is difficult and still results in severely degraded performance when dealing with keywords of different lengths.

6.3.1 *Related work*

In the following, existing approaches to tackle this problem will be discussed. For bio-acoustic event detection, it has been proposed to use multiple models with different sizes [141]. However, when increasing the number of classes this quickly becomes highly impractical. Another approach for bio-acoustic event detection is to use individual frames for classification [131]. While this may work for discriminating between animal calls belonging to different species, which may in fact be very short and often sound very differently, individual frames of short-time cepstral features do not carry enough information to correctly detect spoken keywords. The reason is that very short audio signals containing speech corresponding to individual frames all are very similar to each other if different keywords largely consist of the same phonemes. For [ASR](#) [122], a sequence-to-sequence loss such as the connectionist temporal classification ([CTC](#)) loss [66] is used, which can handle sequences of words with different lengths. However, training such a model requires enormous amounts of data with the same acoustic conditions as the expected data for the application. This problem persists when using pre-trained [ASR](#) systems [97, 145]. Furthermore, using an [ASR](#) system for [KWS](#) leads to a large computational overhead and thus is impractical when needing a very fast inference time or detecting keywords locally on sensors with very limited computational resources. For some applications where only the first instance of a keyword is of interest, computational power can be saved by using spiking neural networks and stopping the search process after finding a single keyword [87]. Still, choosing a fixed input size is required. An approach that can handle arbitrary sizes is to apply dynamic time warping ([DTW](#)) to keyword templates created by concatenating classical short-time cepstral features such as Mel-frequency cepstral coefficients ([MFCCs](#)) or human factor cepstral coefficients ([HFCCs](#)) [116, 222]. For these, the main problem is that the performance quickly degrades in noisy conditions or for short

words. In [146], the same approach was used to collect samples of individual keywords that are later used as training data for an embedding model. Although this may help to improve the performance obtained with the final **KWS** model, it does not solve any of the previously mentioned problems both involved approaches are suffering from. Apart from presenting an embedding-based **KWS** system, the goal of this section is to present a loss function that solves the problem of needing to choose a fixed input size.

6.3.2 Dataset

Conventional **KWS** datasets such as SpeechCommands [231] do not aim at detecting specific keywords in sentences but focus only on classifying isolated words or phrases. Since this eliminates one of the major difficulties of **KWS**, namely the need for detecting on- and offsets of keywords, the few-shot open-set dataset **KWS-DailyTalk** will now be introduced. This dataset is based on the **ASR** dataset DailyTalk [118] containing high-quality speech recordings without noise from scripted conversations. **KWS-DailyTalk** is divided into a training split, a validation split and a test split. The training split of **KWS-DailyTalk** has a length of only 39 s and consists of five isolated samples for each of the following 15 keywords: “afternoon”, “airport”, “cash”, “credit card”, “deposit”, “dollar”, “evening”, “expensive”, “house”, “information”, “money”, “morning”, “night”, “visa” and “yuan”. The validation and test split have a length of approximately 10 min each and consist of 156 and 157 sentences, respectively. Each sentence contains any number of the keywords to be detected including none of them. As a result, each keyword appears roughly 12 times. Their on- and offsets are manually annotated and the training samples stem from other conversations as the sentences contained in the validation and test splits. As an evaluation metric, the micro-averaged F_1 -score (cf. Section 2.10) as implemented in [148] is used. All hyperparameters are set to optimize the performance on the validation split.

6.3.3 System overview

The embedding based few-shot **KWS** system depicted in Figure 27 has a similar structure as the previously used **ASD** systems. First, the waveforms are pre-processed using a frontend. Then embeddings are extracted with a neural network and a backend is applied to compute scores and output the detected keywords. One of the major differences to the previous **OSC** application is that individual instances of different keywords or even the same keyword may have strongly varying lengths. Moreover, a stream of arbitrary length needs to be handled during inference and may contain several acoustic events that are of interest. To be able to do this, the embeddings are designed to have a temporal dimension. All recordings are divided into small segments of a fixed size before computing embeddings with a temporal resolution. After that, the embeddings of individual segments belonging

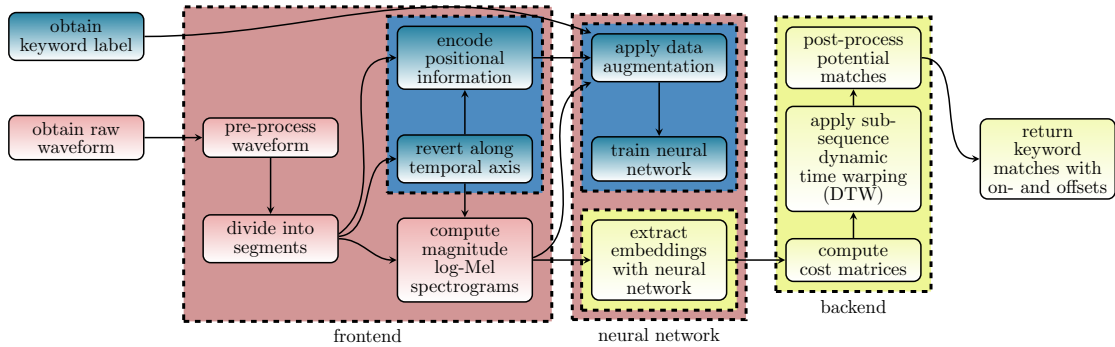


Figure 27: Structure of the proposed **KWS** system. Blocks colored in blue are only used for training the system, blocks colored in yellow are only used for inference and blocks colored in red are used for training and inference. © 2024 IEEE

to the same recording are combined again into a single embedding. Individual keywords are detected by searching their instances provided by the training samples in test recordings using sub-sequence **DTW**. The details of all components will be presented in the following sections.

6.3.4 Extracting embeddings

The frontend applied to obtain input features representations for the embedding model is the following. First, the raw waveforms are re-sampled to 16 kHz, high-pass filtered with a cutoff frequency of 50 Hz and the amplitude is normalized by dividing with the maximum value. For training samples, a segment length $L_{\text{seg}} = 0.25\text{s}$ and an overlap of $\frac{L_{\text{seg}}}{5}$ are used. Any segments shorter than L_{seg} are padded with zeros. During inference, the same segment length in combination with an overlap of $\frac{256}{16000\text{Hz}}$ is used to obtain a higher temporal resolution. Furthermore, samples are padded with $\lfloor \frac{L_{\text{seg}} \cdot 16000}{2} \rfloor$ zeros such that their center aligns with the position in the original recording. After all these pre-processing steps, log-Mel magnitude spectrograms with 64 Mel bins are extracted using a **STFT** with Hanning-weighted windows of size 1024 and a hop size of 256. This results in a temporal dimension of $T := \lfloor L_{\text{seg}} \cdot \frac{16000}{256} \rfloor \in \mathbb{N}$.

The architecture of the embedding model is similar to the one used for **ASD** (see Table 4) and is provided in Table 30. Apart from the differently sized input representations, the differences are that no temporal max-pooling and padding is applied to retain the original temporal dimension of the input data. Furthermore, dropout with a probability of 20% is used after each residual block. The model is trained for 1000 epochs with a batch size of 32 using adam [101] to minimize the temporal AdaCos (**TACos**) loss, which will be presented in Section 6.3.5. During training, random oversampling is applied to balance the number of segments belonging to different keyword classes resulting from the varying lengths

Table 30: Modified ResNet architecture used for extracting embeddings with temporal dimension. © 2024 IEEE

layer name	structure	output size
input	-	16×64
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1×1	$16 \times 64 \times 16$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1×2	$16 \times 32 \times 32$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1×2	$16 \times 16 \times 64$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1×2	$16 \times 8 \times 128$
max pooling	1×8 , stride= 1×1	16×128
dense (embedding)	linear	16×128

of the original keyword recordings. To avoid overfitting, mixup [272] with a uniformly distributed mixing coefficient and SpecAugment [173] are applied for data augmentation purposes. In addition, the background recordings from SpeechCommands [231] are used as additional class “no speech” to reduce the number of false positives detected in segments not containing any speech.

After training, the trained embedding model is used to extract embeddings for the segmented recordings belonging to the entire dataset. Then, embeddings of individual segments are combined by taking the mean of all frames that belong to the same time frame in the original file, resulting in a single embedding with a temporal resolution for the entire file matching the original length in time. This is illustrated in Figure 28.

6.3.5 *TACos*

The idea of the *TACos* loss is to have two training objectives for each segment. First, the correct keyword should be detected with a supervised loss function \mathcal{L}_{kw} . Second, the correct position of a segment within the keyword should be detected with a self-supervised loss \mathcal{L}_{pos} . Without learning the position of segments, the resulting embeddings are often constant for regions where the same keyword is contained because embeddings of segments only need to carry information about the keyword. In these cases, applying DTW is similar to directly classifying individual frames with a classifier and therefore using the embeddings as templates does not significantly improve the performance. Using \mathcal{L}_{pos} as one of the training objectives, forces the embedding model to include positional information into the

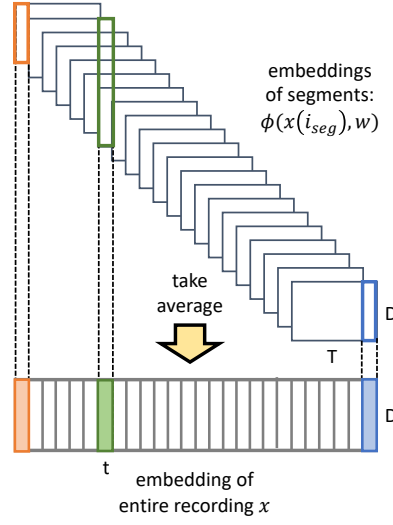


Figure 28: Illustration of combining the embeddings belonging to different segments of a single recording.

embeddings and thus varying the combined embeddings for entire recordings over time.

Processing keyword instances with strongly varying lengths with the same discriminative loss function requires a relative instead of an absolute encoding of the positions, which is defined as follows:

Definition 6.3 (Categorical encoding of relative position). Let $N_{\text{seg}}(\mathbf{x}) \in \mathbb{N}$ denote the number of segments belonging to training sample $\mathbf{x} \in X_{\text{train}}$ and define $N_{\text{pos}} := \max_{\mathbf{x} \in X_{\text{train}}} \{N_{\text{seg}}(\mathbf{x})\}$. For $\mathbf{x} \in X_{\text{train}}$ and segment $i_{\text{seg}} \in \{1, \dots, N_{\text{seg}}(\mathbf{x})\}$, set the interval of active positions to

$$I_{\text{active}}(\mathbf{x}, i_{\text{seg}}) = \left[1 + \left\lfloor \frac{(i_{\text{seg}} - 1) \cdot N_{\text{pos}}}{N_{\text{seg}}(\mathbf{x})} \right\rfloor, \left\lceil \frac{i_{\text{seg}} \cdot N_{\text{pos}}}{N_{\text{seg}}(\mathbf{x})} \right\rceil \right] \subset \mathbb{R}. \quad (67)$$

Then, the *categorical encoding of the relative position* $\text{lab}_{\text{pos}}(\mathbf{x}, i_{\text{seg}}) \in [0, 1]^{N_{\text{pos}}}$ of segment $i_{\text{seg}} \in \{1, \dots, N_{\text{seg}}(\mathbf{x})\}$ belonging to training sample $\mathbf{x} \in X_{\text{train}}$ is defined by setting

$$\text{lab}_{\text{pos}}(\mathbf{x}, i_{\text{seg}})_{j_{\text{pos}}} = \frac{\mathbb{1}_{I_{\text{active}}(\mathbf{x}, i_{\text{seg}})}(j_{\text{pos}})}{\sum_{j_{\text{pos}}=1}^{N_{\text{pos}}} \mathbb{1}_{I_{\text{active}}(\mathbf{x}, i_{\text{seg}})}(j_{\text{pos}})} \quad (68)$$

where $i_{\text{pos}} \in \{1, \dots, N_{\text{pos}}\}$ and $\mathbb{1}_I : \mathbb{R} \rightarrow \{0, 1\}$ denotes the characteristic function for the interval $I \subset \mathbb{R}$.

These positional encodings are used as labels for a self-supervised classification task. Note that for all instances of keywords shorter than the longest one present

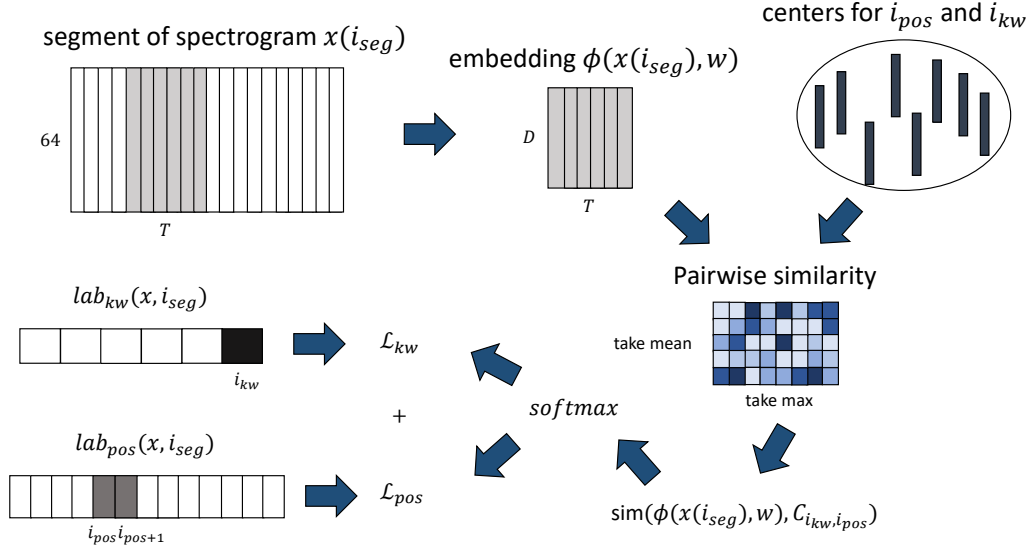


Figure 29: Illustration of the **TACos** loss function. © 2024 IEEE

in the training set, multiple positions of the encoding are set to “active” with equal target probability. The position of segments not containing any speech is encoded with a uniform target distribution meaning that each position of this segment is equally (un)likely.

Now, the actual **TACos** loss function used for training the embedding model will be defined. A graphical illustration of the loss function can be found in [Figure 29](#).

Definition 6.4 (TACos). Using the notation introduced in [Definition 6.3](#), let $Y \subset X$ be finite and let $\mathbf{x}(j) \in \mathbb{R}^{T \times D}$ denote the j -th segment generated from sample $\mathbf{x} \in X$. Furthermore, let $N_{kw} \in \mathbb{N}$ denote the number of keyword classes and let $\mathbf{C}_{i_{kw}, i_{pos}} \in \mathcal{P}(\mathbb{R}^D)$ with $|\mathbf{C}_{i_{kw}, i_{pos}}| = N_{centers}$ denote the trainable centers for keyword $i_{kw} \in \{1, \dots, N_{kw}\}$ and position $i_{pos} \in \{1, \dots, N_{pos}\}$. Define the similarity between embedding $\phi(\mathbf{x}(i_{seg}), \mathbf{w}) = (\phi(\mathbf{x}(i_{seg}), \mathbf{w})_1, \dots, \phi(\mathbf{x}(i_{seg}), \mathbf{w})_T) \in \mathbb{R}^{T \times D}$ and the centers $\mathbf{C}_{i_{kw}, i_{pos}}$ as

$$\begin{aligned} & \text{sim}(\phi(\mathbf{x}(i_{seg}), \mathbf{w}), \mathbf{C}_{i_{kw}, i_{pos}}) \\ & := \text{mean}(\{\max_{\mathbf{c} \in \mathbf{C}_{i_{kw}, i_{pos}}} \text{sim}(\phi(\mathbf{x}(i_{seg}), \mathbf{w})_t, \mathbf{c}) \in \mathbb{R}^D : t = 1, \dots, T\}). \end{aligned} \quad (69)$$

The corresponding softmax probability of embedding $\phi(\mathbf{x}(i_{seg}), \mathbf{w})$ belonging to keyword i_{kw} and position i_{pos} is defined as

$$\begin{aligned} & \text{softmax}(\hat{\mathbf{s}} \cdot \text{sim}(\phi(\mathbf{x}(i_{seg}), \mathbf{w}), \mathbf{C}_{i_{kw}, i_{pos}})) \\ & := \frac{\exp(\hat{\mathbf{s}} \cdot \text{sim}(\phi(\mathbf{x}, \mathbf{w}), \mathbf{C}_{i_{kw}, i_{pos}}))}{\sum_{j_{kw}=1}^{N_{kw}} \sum_{j_{pos}=1}^{N_{pos}} \exp(\hat{\mathbf{s}} \cdot \text{sim}(\phi(\mathbf{x}(i_{seg}), \mathbf{w}), \mathbf{C}_{j_{kw}, j_{pos}}))} \end{aligned} \quad (70)$$

where $\hat{s} \in \mathbb{R}_+$ is the dynamically adaptive scale parameter as defined for the sub-cluster AdaCos loss in 3.4. The probability of embedding $\phi(\chi(\mathbf{i}_{\text{seg}}), \mathbf{w})$ belonging to keyword \mathbf{i}_{kw} is set to $\sum_{\mathbf{j}_{\text{pos}}=1}^{\mathbf{N}_{\text{pos}}} \text{softmax}(\hat{s} \cdot \text{sim}(\phi(\chi(\mathbf{i}_{\text{seg}}), \mathbf{w}), \mathbf{C}_{\mathbf{i}_{\text{kw}}, \mathbf{j}_{\text{pos}}}))$ and the probability of embedding $\phi(\chi(\mathbf{i}_{\text{seg}}), \mathbf{w})$ belonging to position \mathbf{i}_{pos} is set to $\sum_{\mathbf{j}_{\text{kw}}=1}^{\mathbf{N}_{\text{kw}}} \text{softmax}(\hat{s} \cdot \text{sim}(\phi(\chi(\mathbf{i}_{\text{seg}}), \mathbf{w}), \mathbf{C}_{\mathbf{j}_{\text{kw}}, \mathbf{i}_{\text{pos}}}))$. Therefore, the loss functions for a single embedding $\phi(\chi(\mathbf{i}_{\text{seg}}), \mathbf{w})$ are equal to

$$\begin{aligned} \mathcal{L}_{\text{kw}} &: \mathcal{X} \times \{1, \dots, \mathbf{N}_{\text{kw}}\} \times \mathcal{P}(\mathcal{P}(\mathbb{R}^{\mathbf{D}})) \times \Phi \times \mathcal{W} \times \Lambda(\mathbf{N}_{\text{kw}}) \rightarrow \mathbb{R}_+ \\ \mathcal{L}_{\text{kw}}(\chi, \mathbf{i}_{\text{seg}}, \mathcal{C}, \phi, \mathbf{w}, \text{lab}_{\text{kw}}) \\ &:= \sum_{\mathbf{i}_{\text{kw}}=1}^{\mathbf{N}_{\text{kw}}} \text{lab}_{\text{kw}}(\chi, \mathbf{i}_{\text{seg}})_{\mathbf{i}_{\text{kw}}} \log \left(\sum_{\mathbf{i}_{\text{pos}}=1}^{\mathbf{N}_{\text{pos}}} \text{softmax}(\hat{s} \cdot \text{sim}(\phi(\chi(\mathbf{i}_{\text{seg}}), \mathbf{w}), \mathbf{C}_{\mathbf{i}_{\text{kw}}, \mathbf{i}_{\text{pos}}})) \right) \end{aligned} \quad (71)$$

and

$$\begin{aligned} \mathcal{L}_{\text{pos}} &: \mathcal{X} \times \{1, \dots, \mathbf{N}_{\text{pos}}\} \times \mathcal{P}(\mathcal{P}(\mathbb{R}^{\mathbf{D}})) \times \Phi \times \mathcal{W} \times \Lambda(\mathbf{N}_{\text{pos}}) \rightarrow \mathbb{R}_+ \\ \mathcal{L}_{\text{pos}}(\chi, \mathbf{i}_{\text{seg}}, \mathcal{C}, \phi, \mathbf{w}, \text{lab}_{\text{pos}}) \\ &:= \sum_{\mathbf{i}_{\text{pos}}=1}^{\mathbf{N}_{\text{pos}}} \text{lab}_{\text{pos}}(\chi, \mathbf{i}_{\text{seg}})_{\mathbf{i}_{\text{pos}}} \log \left(\sum_{\mathbf{i}_{\text{kw}}=1}^{\mathbf{N}_{\text{kw}}} \text{softmax}(\hat{s} \cdot \text{sim}(\phi(\chi(\mathbf{i}_{\text{seg}}), \mathbf{w}), \mathbf{C}_{\mathbf{i}_{\text{kw}}, \mathbf{i}_{\text{pos}}})) \right) \end{aligned} \quad (72)$$

where $\mathcal{C} \in \mathcal{P}(\mathcal{P}(\mathbb{R}^{\mathbf{D}}))$ with $|\mathcal{C}| = \mathbf{N}_{\text{kw}} \cdot \mathbf{N}_{\text{pos}}$. The TACos loss includes both loss functions and is defined as

$$\begin{aligned} \mathcal{L}_{\text{tac}} &: \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{P}(\mathbb{R}^{\mathbf{D}})) \times \Phi \times \mathcal{W} \times \Lambda(\mathbf{N}_{\text{kw}}) \times \Lambda(\mathbf{N}_{\text{pos}}) \rightarrow \mathbb{R}_+ \\ \mathcal{L}_{\text{tac}}(\mathcal{Y}, \mathcal{C}, \phi, \mathbf{w}, \text{lab}_{\text{kw}}, \text{lab}_{\text{pos}}) \\ &:= - \frac{1}{|\mathcal{Y}|} \sum_{\chi \in \mathcal{Y}} \frac{1}{\mathbf{N}_{\text{seg}}(\chi)} \sum_{\mathbf{i}_{\text{seg}}=1}^{\mathbf{N}_{\text{seg}}(\chi)} \mathcal{L}_{\text{kw}}(\chi, \mathbf{i}_{\text{seg}}, \mathcal{C}, \phi, \mathbf{w}, \text{lab}_{\text{kw}}) + \mathcal{L}_{\text{pos}}(\chi, \mathbf{i}_{\text{seg}}, \mathcal{C}, \phi, \mathbf{w}, \text{lab}_{\text{pos}}). \end{aligned} \quad (73)$$

For all experiments in this section, the hyperparameters $\mathbf{D} = 128$ and $\mathbf{N}_{\text{centers}} = 16$ were used.

To further improve the performance obtained with the KWS system, a second SSL task is used when training the embedding model. More concretely, the model is taught to recognize temporally reversed segments by creating a temporally reversed copy of each segment and creating labels in the following ways: For each keyword class, an additional class for all temporally reversed segments belonging to this keyword class is introduced, almost doubling the total number of classes. The position of temporally reversed segments is encoded with a uniform target distribution as done for the segments not containing any speech. Introducing temporally reversed segments as difficult out-of-distribution samples that still contain speech, makes the training objective much more challenging. This results in more information about the temporal structure of the speech contained in the embeddings and is expected to improve the performance by reducing the number of false alarms.

6.3.6 *DTW backend*

To actually detect appearances of keywords with their on- and offsets, sub-sequence *DTW* is applied in the following way: First, cost matrices are computed using the pairwise cosine distances between the embeddings belonging to test sentences and the embeddings of training samples. Using Fréchet means instead of using the individual samples, obtained by applying the dynamic time warping barycenter averaging (*DBA*) [180], degrades the runtime but also degrades the performance. Since *DTW* can be parallelized by sweeping diagonally over cost matrices, the computational overhead is still manageable and the processing times are much faster than real-time if the number of keywords and shots is not too high. Therefore, the individual samples were used for the experimental evaluations. To accumulate the costs of all cost matrices, the *DTW* step sizes $(1, 1)$, $(1, 2)$ and $(2, 1)$ were used. After this, a warping path is computed for each temporal position and its associated cost is normalized by the length of the corresponding path. Then, the normalized accumulated costs is used as a matching score and a decision threshold, tuned on the validation set, can be applied to find the matches. The on- and offsets of a detected keyword are given by the start and end of the warping paths, whose matching scores are below the decision threshold. For post-processing, overlapping detections are shortened such that they do not overlap and detections shorter than half the duration of the training sample are removed.

6.3.7 *Baseline systems*

To justify the design of the proposed *KWS* system, the following two baseline systems were used for comparison: A classical *HFCC*-based approach and a system using vector-sized embeddings by applying a sliding window. The *HFCC*s are extracted using spectrograms with a window size of 40 ms and a step size of 10 ms. In contrast to the well-known *MFCC*s, *HFCC*s use a triangular filterbank with perceptually motivated bandwidths based on the Bark-scale instead of the center frequencies resulting in filters with less overlap. This was shown to improve the performance for *DTW*-based *KWS* [222]. For the baseline system, the same sub-sequence *DTW* algorithm as used for the trained embeddings is applied.

For the sliding window based system, a slightly modified embedding model architecture as used for the embeddings with the temporal dimension is applied to obtain vector-sized embeddings. One modification is to also apply the max-pooling operation in the residual blocks to the temporal dimension. Another modification is to apply a flattening operation before linearly projecting the hidden representation of the network onto the embedding space. The AdaCos loss is utilized for training the embedding model. To detect keywords with on- and offsets, the following procedure is used: First, the cosine similarities between all temporally sorted embeddings of a target sentence and the keyword-specific centers of the embedding network are computed. Then, these values are compared to a decision threshold

Table 31: Event-based, micro-averaged F-score, precision and recall obtained on **KWS-DailyTalk** with different **KWS** systems. Highest F_1 -scores for each feature representation are highlighted with bold letters, overall highest F_1 -scores are underlined. © 2024 IEEE

feature representation	reversed segments	threshold	obtained performance					
			validation split			test split		
			F ₁ -score	precision	recall	F ₁ -score	precision	recall
HFCCs	not applicable	global	60.52%	63.25%	58.01%	56.97%	61.54%	53.04%
HFCCs	not applicable	individual	64.71%	69.18%	60.77%	57.74%	62.58%	53.59%
embeddings (sliding)	not used	global	39.76%	43.71%	36.46%	38.35%	41.14%	35.91%
embeddings (sliding)	not used	individual	46.96%	49.39%	44.75%	40.44%	40.00%	40.88%
embeddings (sliding)	used	global	44.13%	62.00%	34.25%	44.83%	59.63%	35.91%
embeddings (sliding)	used	individual	55.43%	54.55%	56.35%	50.42%	51.14%	49.72%
embeddings (\mathcal{L}_{kw})	not used	global	56.04%	52.40%	60.22%	54.25%	53.80%	54.70%
embeddings (\mathcal{L}_{kw})	not used	individual	58.38%	57.14%	59.67%	53.04%	53.04%	53.04%
embeddings (\mathcal{L}_{kw})	used	global	64.58%	74.64%	56.91%	61.30%	69.72%	54.70%
embeddings (\mathcal{L}_{kw})	used	individual	66.12%	64.89%	67.40%	60.53%	57.79%	63.54%
embeddings (\mathcal{L}_{tac})	not used	global	62.78%	75.78%	53.59%	64.65%	82.76%	53.04%
embeddings (\mathcal{L}_{tac})	not used	individual	63.36%	63.19%	63.54%	63.31%	68.15%	59.12%
embeddings (\mathcal{L}_{tac})	used	global	65.78%	82.50%	54.70%	70.47%	89.74%	58.01%
embeddings (\mathcal{L}_{tac})	used	individual	69.44%	75.00%	64.64%	69.16%	79.29%	61.33%

resulting in another binary time-series for each keyword containing possible detections. These time-series are post-processed by applying median filters with lengths equal to the nearest odd number of the mean number of frames to all shots of the corresponding keyword. As a result, boxes indicating detections of keywords are obtained, whose start- and endpoints are shifted by $-\frac{L_{seg}}{2}$ and $+\frac{L_{seg}}{2}$, respectively.

6.3.8 Experimental comparison

A comparison of the performance on **KWS-DailyTalk** obtained with different **KWS** systems can be found in **Table 31**. The following observations can be made: First and foremost, the embeddings obtained with the **TACos** loss perform best, which is especially apparent on the test split of the dataset. Second, the embeddings based on a sliding window perform worse while the embeddings with a temporal dimension perform better than classical **HFCCs**. This shows that using a sliding window is highly sub-optimal and justifies utilizing a temporal dimension. Third, using temporally reversed embeddings for training improves the performance for all embeddings. Since this simple **SSL** approach is applied in addition to other powerful data augmentation techniques such as mixup and SpecAugment, this shows that doing so is highly beneficial. Last but not least, individually tuned decision thresholds do not improve the performance of the embeddings with a temporal dimension on the test split. Therefore, an extensive tuning of decision thresholds is not necessary and can be omitted.

6.4 SUMMARY

In this chapter, **OSC** and **SED** were investigated as additional **ASD** applications in few-shot settings, meaning that only very few training samples were provided. For **OSC**, a dataset for acoustic alarm detection in domestic environments and, for **SED**, a **KWS** application were considered. Similar to the results presented in [Section 5.5](#), it was shown that directly training a model for the few-shot **OSC** task leads to better results than using pre-trained embeddings. For a sufficiently high number of shots used for training, the results obtained with the presented system were close to optimal performance. However, the gap in performance decreased the less training samples were available and for an openness equal to zero, i.e. a **CSC** task, the pre-trained embeddings performed slightly better.

For the **KWS** task, a novel loss function, named **TACos**, and a few-shot **KWS** dataset were presented. The **TACos** loss aims at learning embeddings with a temporal structure and consists of a supervised task for predicting the keyword and a self-supervised task for predicting the position of a short audio segment. Additionally, a few-shot **KWS** system utilizing these embeddings as features for **DTW** was presented. Utilizing these embeddings has the advantage that the system is able to detect keywords of strongly varying lengths when monitoring audio streams because a fixed window size for processing the data is not necessary. In experimental evaluations, it was shown that also predicting the position when training the embedding model significantly improves the performance because the model is forced to learn embeddings that change over time, which makes them much better suited as features for **DTW**. As a result, it was shown that the performance of the proposed **KWS** system is better than the performance obtained with models based on hand-crafted speech features or when using an embedding model based on a sliding window. Furthermore, an **SSL** approach using temporally reversed segments as negative examples during training was proposed and it was shown that this approach improves the performance regardless of the model.

CONCLUSION

7.1 SUMMARY

Reliably detecting anomalous sounds is important for various applications ranging from monitoring machines for predictive maintenance, entire acoustic scenes for security or the health of persons to acoustic open-set classification where anomalies correspond to unknown classes. However, training an automatic ASD system is difficult because for most applications anomalous samples are not available for training. Additional challenges in a semi-supervised setting are the high dimensionality of audio data and unwanted variations resulting from changing properties of the sensors or the sound sources themselves. Furthermore, there is no inherent property that is different for normal and anomalous sounds as defining these terms entirely depends on the application. To solve all these issues, the goal of this thesis was to investigate how to obtain a mapping from the audio signal space to a relatively low-dimensional vector space, called embedding space, where normal and anomalous sounds can be distinguished easily. For the experimental evaluations, acoustic machine condition monitoring served as the main application throughout the thesis.

First, the structure of a state-of-the-art ASD system based on audio embeddings was presented by reviewing the existing literature. Such a system consists of a frontend, an embedding model and a backend. In the frontend, the audio signals are pre-processed and their dimension is reduced by converting them into time-frequency representations. Depending on the training objective, there are three different types of embedding models, each with different strengths and weaknesses. The model can be trained by using one-class losses such as the reconstruction or the compactness loss, by solving auxiliary classification tasks with angular margin losses based on provided meta information or SSL, or by using models pre-trained on large datasets. To improve the performance, data augmentation techniques and ensembling can be applied. The goal of the backend is to compute an anomaly score by measuring the distance to normal embeddings or estimating their distribution and computing the likelihood of test embeddings. In addition to discussing the structure of an ASD system, threshold-dependent and threshold-independent evaluation metrics that measure the performance of a system as well as methods for estimating a decision threshold were discussed.

In the third chapter, one-class embeddings were compared to auxiliary task embeddings. To this end, it was shown that learning a joint embedding space by utilizing a classification task, which incorporates as much meta information as possible, yields much better results than one-class models. The reason is that solving a classification task enables the embedding model to closely monitor target sounds

and ignore the background noise resulting in a stronger sensitivity to anomalous deviations from normal sounds, which may be very subtle. As a theoretical result, it was proven that the AdaCos loss minimizes intra-class compactness losses while also maximizing inter-class compactness losses. This shows that both loss functions are strongly related and explains why angular margin losses lead to a good performance. Furthermore, the sub-cluster AdaCos loss was presented. It generalizes the AdaCos loss by using multiple centers for each class to learn less restrictive distributions leaving more space for anomalous samples between the normal samples. The theoretical results obtained for the AdaCos loss were also extended to the sub-cluster AdaCos loss. An ensemble using this loss function as well as mixup for training and a [GMM](#) as a backend reached a new state-of-the-art performance on the DCASE2020 dataset with an [AUC-ROC](#) of 97%, which is close to optimal.

When using an [ASD](#) system in a specific application, decision thresholds need to be specified to decide between normal and anomalous samples. Different methods for estimating a decision threshold from normal data only were compared experimentally. It was shown that most methods lead to similar results that lie between 90% and 95% of the performance obtained with an optimal decision threshold and that multi-stage approaches perform slightly better. Furthermore, holding back samples to obtain more realistic anomaly scores for estimating a threshold did not improve performance and thus is not needed. To also include the difficulty of estimating a good decision threshold, which is not captured by the [AUC-ROC](#) score, the threshold-independent evaluation metric F_1 -[EV](#) was presented. In experimental evaluations, it was shown that this metric has a high correlation with the [AUC-ROC](#) score as well as with the F_1 score and thus has the potential to replace the [AUC-ROC](#) score as the standard metric for semi-supervised [ASD](#).

The fifth chapter dealt with domain generalization for [ASD](#), i.e. designing the system in such a way that it is robust to potential changes of the distribution of normal data. The designed system consists of two sub-networks using [DFT](#) and [STFT](#) based features as input, concatenating the resulting embeddings and calculating the cosine distance as an anomaly score. Trivial solutions for individual classes were prevented by not using any bias terms and non-trainable class centers. By visualizing the embedding space using [t-SNE](#) and utilizing [RISE](#) to explain the decisions of the proposed system, it was shown that learning a joint embedding space with multiple classes helps to capture more relevant information with the embeddings than using multiple embedding models. In comparison to systems based on pre-trained embeddings, the performance obtained with the system was significantly better despite having only very few samples for the target domains. To further improve the performance of the system, the AdaProj loss was proposed. This loss generalizes the sub-cluster AdaCos loss by learning arbitrary distributions in linear sub-spaces for each class. Additionally, it was shown that applying [SSL](#) approaches in the form of [StatEx](#) and a novel [FeatEx](#) approach, which randomly exchanges the embeddings of the two sub-networks, significantly improves the performance. An ensemble based on multiple presented systems outperformed all

other published systems on the DCASE2023 dataset by a large margin and thus reached a new state-of-the-art performance.

To cover a broader range of **ASD** applications than acoustic machine condition monitoring, few-shot **OSC** and few-shot **KWS** applications were investigated. It was shown that the previously developed **ASD** system also performs well for few-shot **OSC** and that this system outperforms systems using pre-trained embeddings even in case only very few training samples are available for every normal class. For few-shot **KWS**, **TACos**, a novel loss function was presented. Here, embeddings with a temporal resolution are learned that can be used as templates for **DTW** to effectively deal with varying lengths of keywords. Besides the supervised task of predicting the correct keyword a short segment of a spectrogram belongs to, **TACos** utilizes two **SSL** approaches: Namely, also predicting the position of each segment inside a keyword sample and detecting temporally reversed segments as challenging negative samples. In experiments on a novel few-shot **KWS** dataset, it was shown that both **SSL** approaches significantly improve the performance and that a system utilizing the **TACos** loss outperforms systems that classify individual segments using a sliding window or use hand-crafted speech features such as **HFCCs**.

7.2 OUTLOOK AND FUTURE WORK

As stated in the previous section, many improvements for designing a semi-supervised **ASD** system based on audio embeddings were achieved. Still, the problem is far from being solved, especially in challenging conditions, and thus is expected to be an interesting research question for many years to come. In the following, possible directions for future research will be discussed.

One possible direction is to further improve the performance of **ASD** systems for acoustic machine condition monitoring by finding more efficient methods to handle the noise and non-target sound events other than using meta information, which may not always be available. Apart from using additional **SSL** approaches, applying source separation to isolate the machine sounds is a very promising approach [22, 100, 178, 203]. Although it is very difficult to achieve, one could aim for separating signals into three source: One source containing the noise and non-target sound components, another source containing the normal signal components and a third source containing the anomalous signal components. Reaching this goal would not only lead to much better performance but also greatly improve the explainability of the results, which is an important goal on its own. An additional way to improve the performance may be to investigate more sophisticated methods than randomly initializing the centers of the presented angular margin loss functions. Furthermore, the methods presented in this thesis can be applied to strongly related fields such as time series analysis [120] or structural health monitoring [13, 14] to exchange ideas developed in different research communities. Extensive knowledge of subject matter experts working in different fields may also

help to introduce additional evaluation metrics such as the presented F_1 -*EV* score that capture all practical demands placed upon an *ASD* systems.

Another topic for future research is to extend the presented systems for few-shot *SED* to zero-shot learning [117]. This means that one is interested in detecting sound events belonging to classes for which no audio samples can be provided, which enables users to search for arbitrary sound events. State-of-the-art zero-shot learning systems are taught to project pre-trained embeddings of an audio embedding space and a second embedding space utilizing another data type into a joint embedding space. This allows to compare the embedding of an arbitrary audio recording to pre-trained embeddings obtained with textual input [49, 263] or images [41]. In [130], so-called sound attribute vectors describing the sounds were introduced that enable users to define new sound classes by creating corresponding sound attribute vectors. To obtain general-purpose audio embeddings, the pre-trained models presented in Section 2.5 as well as more specialized embeddings such as wav2vec embeddings [8] or other embeddings pre-trained on a large *ASR* dataset [18, 103] can be used.

As a third research direction, the semi-supervised *ASD* setting investigated in this thesis can be extended to detecting anomalous sound events in a continuous audio stream, which can be seen as a combination of *ASD* and *SED*. The difference to *ASD* is that anomalous sounds need to be localized in time and not only be provided as results for isolated sounds. The difference to *SED* is that one does not have any training data for the anomalous sound events that are of actual interest. In conclusion, solving this task appears to be difficult. Still, in any monitoring application one usually encounters continuous audio streams and the interesting events needing immediate attention are the ones that are anomalous. Furthermore, developing such a system could also be used for active learning of unknown sound events [204, 229, 275].

A

APPENDIX

A.1 KEY PUBLICATIONS

In this section, the key publications that contain substantial parts of this thesis are attached for inspection of the doctoral committee. For each publication, individual contributions of co-authors are stated explicitly. The remaining content of each publication is the sole contribution of the thesis author. Publications protected by the copyright of IEEE are subject to the following: *In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of Bonn's products or services. Internal or personal use of this material is permitted. If interested in reprinting/re-publishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.*

A.1.1 *Key publication 1*

Kevin Wilkinghoff. "Sub-Cluster AdaCos: Learning Representations for Anomalous Sound Detection." In: *International Joint Conference on Neural Networks*. IEEE, 2021. DOI: [10.1109/IJCNN52387.2021.9534290](https://doi.org/10.1109/IJCNN52387.2021.9534290).

© 2021 IEEE.

Sub-Cluster AdaCos: Learning Representations for Anomalous Sound Detection

Kevin Wilkinghoff

Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE

Fraunhoferstraße 20, 53343 Wachtberg, Germany

kevin.wilkinghoff@fkie.fraunhofer.de

Abstract—When training a model for anomalous sound detection, one usually needs to estimate the underlying distribution of the normal data. By doing so, anomalous data has a lower probability in view of this distribution than normal data and thus can easily be detected. However, audio data is very high-dimensional making it difficult to have a good estimate of the true distribution. To have more accurate estimates, the dimension of the data can be reduced first. One way to do this is to train discriminative neural networks for extracting lower-dimensional representations of the data. Particularly, neural networks trained with angular margin losses as AdaCos have been shown to perform well for this task. In this work, a modified AdaCos loss called sub-cluster AdaCos specifically designed for detecting anomalous data is presented. In multiple experiments conducted on the DCASE 2020 dataset for “Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring”, these design choices are empirically justified. As a result, a conceptually simple system for anomalous sound detection is presented that significantly outperforms all other published systems on this dataset.

Index Terms—machine listening, anomaly detection, representation learning, angular margin loss

I. INTRODUCTION

Anomalous sound detection has various applications such as detecting traffic accidents in road surveillance systems [1], [2], detecting terrorist attacks in subway stations [3] or machine condition monitoring [4]. Furthermore, all open-set classification problems include anomaly detection as a subtask since not all classes are known *a priori* when training the system. Hence, anomalous sound detection is of special interest for many machine listening applications. One can distinguish three major branches of anomaly detection: supervised, semi-supervised and unsupervised [5]. For supervised anomaly detection, two labeled datasets consisting of normal data and anomalous data are used. Although the space of anomalies is huge because it consists of everything that is not considered normal and thus can never be captured exhaustively, samples of anomalies can simplify the training process. This is especially true, when all expected anomalies sound roughly alike as for example when detecting accidents with traffic surveillance systems. In contrast to unsupervised anomaly detection where the training dataset can also include anomalous data and it is not known whether a data sample is normal or not, semi-supervised anomaly detection consists of a clean training dataset containing only normal data. Note that in most practical applications it is much easier to collect

normal data than anomalous data. The main reasons are that anomalous events occur only rarely and can sound much more diverse than recordings of the normal condition. Using the traffic example again, large amounts of audio data belonging to regular road traffic can be collected much easier than recordings of accidents i.e. anomalous data. When considering terrorist attacks, this is even more evident. Therefore, not needing anomalous data for training a system is more suitable for realistic applications and thus a semi-supervised setting is considered in this work.

Usually, semi-supervised or unsupervised anomaly detection boils down to estimating the true underlying distribution of the known data without the aid of sample outliers. Afterwards, one can utilize this distribution to compute the log-probability of the test data or an approximation of it and decide whether this data is an inlier or an outlier using a threshold. However, raw data e.g. waveforms is mostly very high-dimensional making it challenging to estimate the corresponding distribution with limited training data resources. To circumvent this problem, one can train a model for extracting suitable, lower-dimensional representations of the data, which capture enough information such that representations belonging to outliers strongly deviate from inliers. One way to train such a model is to discriminate among all known classes in a supervised manner and utilize the output of an intermediate layer as a feature extractor for a lower-dimensional representation of the data. The assumption is that in order to be able to discriminate among the known classes, all representations need to capture enough information of the raw data and this information is also sufficient to detect outliers. In particular, angular margin losses such as ArcFace [6] and AdaCos [7] have been shown to work well for this task. The reason is that they enforce a low intra-class variability and a high inter-class variability by minimizing the cosine angle of a known class to a learned mean value and ensuring a margin for angles between different classes.

The aim of this work is to investigate specific design choices for an anomalous sound detection system based on an angular margin loss function. The contributions are the following: First, several changes for the AdaCos loss function are proposed leading to a novel loss function called sub-cluster AdaCos. The proposed changes are 1) taking into account the use of mixup to augment the samples, 2) utilizing sub-clusters to learn a less restrictive distribution than standard

AdaCos and 3) using Gaussian distributions or more generally Gaussian mixture models (GMMs) as a backend. In various experiments, it is shown that all of these changes lead to significant improvements in performance when detecting anomalous sounds. As a result, a conceptually simple anomalous sound detection system is presented that significantly outperforms all other published systems on the DCASE 2020 dataset for “Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring” [4].

II. RELATED WORK

Recent work on machine listening is heavily promoted through the annual “Detection and Classification of Acoustic Scenes and Events (DCASE) Workshop” and the associated challenges. Anomalous sound detection is not an exception. Of particular interest for this work is task 2 “Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring” [4] of DCASE 2020. The baseline system of this task consists of class-dependent autoencoders for encoding and decoding consecutive frames of log-Mel spectrograms and utilizing the reconstruction error as an anomaly score. The underlying assumption is that normal data, which has also been used for training the autoencoders, can be reconstructed much better than anomalous data. This fundamental approach of using autoencoders for detecting anomalous data has been extended by utilizing autoencoders conditioned on the machine ids i.e. the class labels [8], [9]. The main idea is that a single autoencoder is used instead of a separate one for each class and trained to have a low reconstruction error when being conditioned on the correct class and a high reconstruction error when being conditioned on another class.

A completely different approach is to train a neural network to discriminate among all known classes. By essentially treating the other classes as anomalous data, decision boundaries are learned for the normal data of each class. Several systems following this approach have been developed independently in the DCASE challenge 2020, most of which use neural networks with a suitable loss function that also reduces intra-class variability to extract lower-dimensional representations of the data. Inoue et al. [10] use center loss [11], Lopez et al. [12] an additive margin softmax layer [13] and Giri et al. [14] as well as Zhou [15] use ArcFace [6] for this purpose. After training such a discriminative network, these lower dimensional representations are utilized in different ways to obtain anomaly detection scores. In most cases, the direct output of the trained representation model or the cosine similarity are used. Another method is to train an additional backend model as for example probabilistic linear discriminant analysis (PLDA) [16] as done in [17].

III. METHODOLOGY

The purpose of this section is to first give a short review on angular margin losses, particularly AdaCos, and then propose a modified AdaCos loss that results in significantly better anomalous sound detection performance.

A. Standard AdaCos loss function

For many years the softmax function in combination with the categorical crossentropy as a loss function has been the standard output layer for classification tasks solved by neural networks. To avoid any confusion, within this work the term “class” corresponds to one of the known classes for which normal training data is available. This means that each machine id is treated as another class. But when training a neural network for the purpose of extracting lower-dimensional representations of the data, so-called embeddings, the softmax function only leads to representations that are linearly separable without explicitly reducing intra-class and increasing inter-class distance of samples. To address this issue, losses based on the Euclidean distance as for example triplet loss [18] and center loss [11] were proposed. Triplet loss uses an anchor input whose distance to a positive and a negative input sample belonging to the same and to another class is minimized and maximized, respectively. Center loss avoids constructing these triplets as input by minimizing the distance to learned center vectors for each class. Recently, angular margin loss functions such as ArcFace [6] have been shown to have better generalization capabilities than losses based on the Euclidean distance by enforcing a margin between angles of samples belonging to different classes. However, the performance in a particular task obtained with angular margin losses heavily relies on fine-tuning their hyperparameters, namely the scale parameter s and the angular margin parameter m . Therefore, AdaCos [7], which uses an adaptive scale parameter and does not depend on any manually set hyperparameters, was developed. Since both losses, ArcFace and AdaCos, lead to similar performance and tuning additional parameters is time consuming, this work focuses entirely on AdaCos.

Let us now formally introduce AdaCos. For AdaCos, the probability of sample $x_i \in \mathbb{R}^D$ belonging to class j of the $C \in \mathbb{N}$ classes is given by

$$P_{i,j} := \frac{\exp(\tilde{s} \cdot \cos \theta_{i,j})}{\sum_{k=1}^C \exp(\tilde{s} \cdot \cos \theta_{i,k})} \quad (1)$$

where $\theta_{i,j} \in [0, \pi]$ is defined through the cosine similarity $\cos \theta_{i,j} = \langle x_i, W_j \rangle / \|x_i\| \|W_j\|$ for a learned class center $W_j \in \mathbb{R}^D$. The dynamically adaptive scale parameter $\tilde{s}^{(t)}$ at training step $t \in \mathbb{N}_0$ is defined as

$$\tilde{s}^{(t)} := \begin{cases} \sqrt{2} \cdot \log(C-1) & \text{if } t = 0 \\ \frac{\log B_{\text{avg}}^{(t)}}{\cos(\min(\frac{\pi}{4}, \theta_{\text{med}}^{(t)}))} & \text{else} \end{cases} \quad (2)$$

where $\theta_{\text{med}}^{(t)} \in [0, \pi]$ denotes the median of all angles θ_{i,y_i} belonging to a mini-batch of size $N \in \mathbb{N}$ with y_i being the class of x_i and

$$B_{\text{avg}}^{(t)} := \frac{1}{N} \sum_{i \in \mathcal{N}^{(t)}} \sum_{\substack{k=1 \\ k \neq y_i}}^C \exp(\tilde{s}^{(t-1)} \cdot \cos \theta_{i,k}) \quad (3)$$

with $\mathcal{N}^{(t)}$ denoting all indices of the samples belonging to the mini-batch of size $N \in \mathbb{N}$.

B. Sub-Cluster AdaCos loss function

The data augmentation technique ‘‘mixup’’ [19] is known to significantly improve the classification performance, since overfitting of a model to specific training data is prohibited. This is done by linearly interpolating between two samples $x_i, x_j \in \mathbb{R}^D$ contained in a mini-batch and their corresponding one-hot encoded class labels $\bar{y}_i, \bar{y}_j \in [0, 1]^C$

$$\begin{aligned} x^{\text{mixed}} &:= \lambda x_i + (1 - \lambda)x_j \\ \bar{y}^{\text{mixed}} &:= \lambda \bar{y}_i + (1 - \lambda)\bar{y}_j \end{aligned} \quad (4)$$

where the mixing coefficient $\lambda \in [0, 1]$ is drawn at random from a suitable distribution. But if one uses mixup, standard AdaCos is not well-defined since most mixed-up samples do not belong to a single class but multiple ones and thus do not have a well-defined class mean. Even when both mixed-up samples belong to the same class and thus have a well-defined class mean, these samples can be treated as anomalies when updating the AdaCos parameters to have a sharper boundary around the support of the distribution of the normal, non-mixed samples of this particular class. To incorporate these changes into the AdaCos function, only mixed-up samples are used for training with mixing coefficients drawn from the uniform distribution. Furthermore, $\hat{\theta}_{\text{med}}^{(t)}$ is the median of the mixed-up angles and $\hat{B}_{\text{avg}}^{(t)}$ is computed using all angles present in a mini-batch, not only the angles of the non-corresponding classes. The details will follow below.

When using AdaCos, only a single cluster is formed for each class and this enforces a single Gaussian distribution for the learned representations after projecting them to the unit sphere by normalizing their lengths. However, anomalous data is easier to detect when a more general, less restrictive distribution is learned for the representations. Gaussian mixture models can approximate any given smooth probability density function. Therefore, a canonical choice to relax the restriction on the distribution imposed by the AdaCos loss is to allow multiple sub-clusters. The same idea of automatically finding subclasses is also used in subclass discriminant analysis and has been shown to outperform other discriminant analysis approaches without subclasses [20], [21]. Note that Deng et al. proposed a very similar approach for the ArcFace loss, namely sub-center ArcFace that uses multiple learned sub-centers for each class instead of only one to efficiently handle label noise present in the training data [22]. This is done by first training the model and then dropping all samples belonging to small sub-clusters, so called non-dominant sub-centers. However, there is a subtle but important difference between sub-center ArcFace and sub-cluster AdaCos. For sub-center ArcFace, only the closest sub-center is considered by taking the maximum cosine similarity. Here, all softmax scores of all sub-clusters of any given class are later summed up to encourage the usage of multiple sub-clusters and thus modeling a more complex distribution.

To avoid numerical issues of large arguments inside the exponential function, which frequently arose in the conducted experiments due to the changes from above, a re-scaling trick that is also used in many implementations of the softmax

function is applied. More concretely, the maximum value of the logits is determined

$$f_{\text{max}}^{(t)} := \max_{i \in \mathcal{N}^{(t)}} \max_{j=1}^{CS} (\hat{s}^{(t-1)} \cdot \cos \theta_{i,j}) \quad (5)$$

where $S \in \mathbb{N}$ denotes the number of sub-clusters and inserted appropriately into the formulas for $\hat{B}_{\text{avg}}^{(t)}$ and $\hat{s}^{(t)}$.

Using the same notation as before, Sub-Cluster AdaCos with all proposed changes is now defined as follows. The modified adaptive scale parameter $\hat{s}^{(t)}$ is defined as

$$\hat{s}^{(t)} := \begin{cases} \sqrt{2} \cdot \log(CS - 1) & \text{if } t = 0 \\ \frac{f_{\text{max}}^{(t)} + \log \hat{B}_{\text{avg}}^{(t)}}{\cos(\min(\frac{\pi}{4}, \hat{\theta}_{\text{med}}^{(t)}))} & \text{else} \end{cases} \quad (6)$$

where

$$\hat{B}_{\text{avg}}^{(t)} := \frac{1}{N} \sum_{i \in \mathcal{N}^{(t)}} \sum_{k=1}^{CS} \exp(\hat{s}^{(t-1)} \cdot \cos \theta_{i,k} - f_{\text{max}}^{(t)}) \quad (7)$$

and $\hat{\theta}_{\text{med}}^{(t)}$ is the median of the mixed-up angles $\theta_{k,y_k}^{\text{mixed}}$. For a mixed-up sample $x_k^{\text{mixed}} = \lambda x_i + (1 - \lambda)x_j$ of the mini-batch with λ fixed, this means

$$\theta_{k,y_k}^{\text{mixed}} := \lambda \theta_{i,y_i} + (1 - \lambda)\theta_{j,y_j}. \quad (8)$$

Finally, the probabilities of sample x_i belonging to class j are given by

$$\hat{P}_{i,j} := \sum_{l \in \mathcal{M}^{(j)}} \frac{\exp(\hat{s} \cdot \cos \theta_{i,l})}{\sum_{k=1}^{CS} \exp(\hat{s} \cdot \cos \theta_{i,k})} \quad (9)$$

where $\mathcal{M}^{(j)}$ denotes all sub-clusters belonging to class j .

C. Outlier detection backend

When training a neural network to extract representations for anomaly detection, there are multiple ways of how to obtain scores and reach a final decision. Now, by using an angular margin loss, the most natural choice to calculate a score besides using the model itself, is to use the cosine similarity of a sample to the center of the corresponding class. A Gaussian distribution can also be seen as an alternative version of the cosine similarity, since the representations are trained such that they are scattered around a learned mean value after projecting them onto the unit sphere by normalizing their lengths, and the unit sphere is locally Euclidean as a manifold. Moreover, Gaussian distributions are more general than the cosine similarity as a scoring method because the cosine similarity is equivalent to using a Gaussian distribution with a spherical covariance matrix i.e. a diagonal matrix with all entries being equal. This is not a problem when doing closed-set classification because only the closest class mean is important. But for anomaly detection, this assumption does not need to be true as illustrated in Fig. 1. Therefore, a Gaussian distribution with full covariance matrix can more accurately estimate the distribution of the normal samples, in case it slightly deviates from a spherical distribution. This makes the detection of anomalies slightly more robust and thus a full covariance matrix is more suitable for the task. When using sub-cluster AdaCos, one can use a GMM with multiple modes

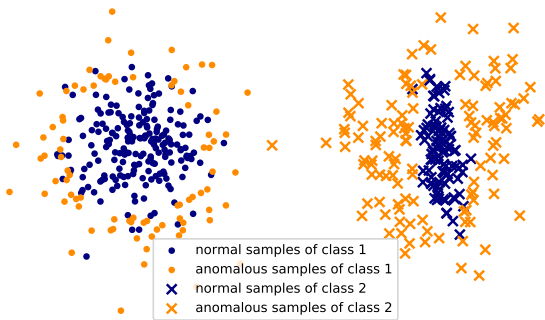


Fig. 1. Scatter plot of normal and anomalous data belonging to two different classes after projecting 3 dimensional representations onto the unit sphere and locally embedding the unit sphere into the 2 dimensional space. Both classes can be easily separated by measuring the distance to the mean. For class 1, anomalies can also be detected reasonably well, but for class 2 only measuring the distance to the mean does not work well because the data is not distributed spherically. In this case, a Gaussian with full covariance matrix would perform much better. Note that this plot is exaggerated for illustration purposes because an angular margin loss ensures that the distribution for each class is roughly spherical. Still, the distribution can slightly deviate from that making a full covariance matrix more suitable, especially in higher-dimensional spaces.

equal to the number of sub-clusters and initialize the means of the modes as the learned sub-cluster centers. After training, the highest log-probability among all the modes can be utilized as an anomaly detection score.

IV. EXPERIMENTAL RESULTS

A. Dataset

For all experiments in this work the dataset belonging to task 2 “Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring” [4] of the DCASE challenge 2020 is used. As the name already implies, the task is to tell whether a recording is normal i.e. belongs to a fully functioning machine or not. The dataset consists of Wav files, each of length 10s and a sampling rate of 16kHz. Each file belongs to one of six different machine types, namely “fan”, “pump”, “slider”, “valve” from MIMII [23] and “ToyCar” and “ToyConveyor” from ToyADMOS [24]. For each of the 6 machine types there are 7 different machine ids except for “ToyConveyor”, which has recordings from 6 different machines. When training and testing it is known to which of the total 41 machines a given audio file belongs to. Furthermore, the dataset is divided into a training set, only consisting of around 1000 normal samples of all machine ids, and a development set and an evaluation set, both consisting of a few hundred normal and anomalous samples from mutually exclusive sets of 3 or 4 machine ids for each machine type. It is not allowed to use any of the files from the development or evaluation set for training. Hence, only normal data is available to train the system and it is known that the training dataset consists of normal samples only. Therefore, despite its name, the anomaly detection task is actually semi-supervised instead of unsupervised. The evaluation metrics for the dataset

TABLE I
MODIFIED RESNET ARCHITECTURE USED FOR ALL EXPERIMENTS.

layer name	structure	output size
input	-	313×128
2D convolution	7×7 , stride= 2	$157 \times 64 \times 16$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$78 \times 31 \times 16$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$39 \times 16 \times 32$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$20 \times 8 \times 64$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$10 \times 4 \times 128$
max pooling	10×1 , stride= 1	4×128
flatten	-	512
dense (representation)	linear	128
AdaCos	-	41

are the area under the receiver operating characteristic (ROC) curve (AUC) and the partial AUC (pAUC) with $p = 0.1$. The metric pAUC is the AUC computed under a low false-positive-rate range, namely $[0, p]$, which is used because a high true-positive-rate is desirable under this conditions in practical applications to avoid frequent false alarms. For more details about the dataset, the reader is referred to [4].

B. Input features and neural network architecture

Using the dataset described above a neural network architecture for extracting lower-dimensional representations of the data as well as their input features need to be defined. Doing so is the purpose of this section. First, the raw waveforms are converted into log-Mel spectrograms to initially reduce their dimension. More concretely, log-Mel spectrograms with 128 Mel bins, a window size of 1024 and a hop size of 512 are computed, which results in a time dimension of 313. Before inserting them as features into the neural network, all log-Mel spectrograms are normalized by subtracting the temporal mean and dividing by the temporal standard deviation of all files belonging to the training dataset. It has also been experimented with using openL3 embeddings [25] as intermediate representations instead of directly using the log-Mel spectrograms as done in [17], [26], but this degraded the performance.

The network architecture used in this work is a modified version of the ResNet architecture [27] with only a few layers and extracts 128-dimensional representations of the data. In each residual block, the leaky ReLU activation function [28] and batch normalization [29] are used. A detailed description of the model can be found in Tab I. It is worth noting that increasing the number of sub-clusters S used for AdaCos also increases the number of parameters significantly. A model without sub-clusters has 772,192 trainable parameters and when using 64 sub-clusters, which results in the largest model used in the conducted experiments, this number increases to 1,102,816. Still, the number of parameters is relatively small. When training the model, the training data belonging to all 41 machine ids is used. Using L^2 -regularization applied to the

TABLE II
MEAN AUCs AND PAUCs OBTAINED WITH DIFFERENT BACKENDS ON THE DEVELOPMENT AND EVALUATION SET.

backend	development set		evaluation set	
	AUC	pAUC	AUC	pAUC
representation model output	87.20%	81.70%	89.55%	83.79%
cosine similarity to mean	88.71%	82.12%	91.13%	84.40%
cosine similarity to top 10	88.69%	82.12%	91.10%	84.38%
two-covariance PLDA	88.25%	82.18%	90.90%	84.32%
Gaussian (spherical covariance)	88.69%	82.12%	91.11%	84.38%
Gaussian (diagonal covariance)	88.71%	82.16%	91.12%	84.39%
Gaussian (full covariance)	89.13%	82.59%	91.43%	84.47%

weights, the model is trained for 400 epochs with a batch-size of 64 to discriminate among these classes by minimizing different versions of the AdaCos loss with Adam [30] and is implemented in Tensorflow [31]. Unless stated otherwise, “mixup” [19] with a mixing coefficient drawn from a uniform distribution and no other data augmentation technique is used.

C. Comparison of different backends

First, the performance obtained with different backends will be compared. For that purpose, a neural network with the standard AdaCos loss and using mix-up is trained to extract the representations. Using the same extracted embeddings, multiple backends are evaluated: the output of the model itself, the cosine similarity to the mean of each machine id, the mean of the cosine similarities to the 10 closest representations from the training set belonging to the same machine id, the log-likelihood ratios of a two-covariance PLDA model as implemented in [32] and Gaussian distributions, each trained for a single machine id, with a spherical, diagonal or full covariance matrix as implemented in scikit-learn [33]. The results can be found in Tab. II.

There are four observations to be made. First, the direct angular softmax output of the model performs significantly worse than all other scoring techniques. Second, PLDA performs better than the direct output but still worse than the remaining backends. Third, as expected, the cosine similarity based backends and the Gaussians with spherical or diagonal covariance matrix all lead to very similar results supporting the claim from before that they are equivalent. And fourth, a Gaussian with a full covariance matrix outperforms all other backends. Hence, in all remaining experiments only Gaussians with full covariance matrix will be used as backends.

D. Interplay of mixup and AdaCos

Next, it is investigated whether including mixup for training the model and the proposed changes of the AdaCos loss function to properly work with mixup improve the performance. The results are depicted in Tab. III. One can see that the performance decreases significantly, when not using mixup, even when the standard AdaCos loss is used as it is. Furthermore, the proposed changes to the AdaCos loss lead to significant improvements in terms of AUC and pAUC on the development set. For the evaluation set, AUC slightly increases and pAUC slightly decreases. A possible explanation for this

TABLE III
MEAN AUCs AND PAUCs OBTAINED WITH MIXUP AND THE MODIFIED ADACOS LOSS, BUT WITHOUT USING SUB-CLUSTERS, ON THE DEVELOPMENT AND EVALUATION SET.

mixup	modified AdaCos	development set		evaluation set	
		AUC	pAUC	AUC	pAUC
		86.96%	80.68%	89.63%	82.47%
	\times	diverges	diverges	diverges	diverges
\times		89.13%	82.59%	91.43%	84.47%
\times	\times	91.60%	85.01%	91.64%	83.93%

TABLE IV
MEAN AUCs AND PAUCs OBTAINED WITH THE MODIFIED SUB-CLUSTER ADACOS LOSS ON THE DEVELOPMENT AND EVALUATION SET.

number of sub-clusters	backend	development set		evaluation set	
		AUC	pAUC	AUC	pAUC
1	Gaussian	91.60%	85.01%	91.64%	83.93%
2	Gaussian	90.97%	82.54%	92.08%	85.08%
4	Gaussian	91.54%	83.53%	92.62%	84.31%
8	Gaussian	91.61%	85.24%	92.99%	85.74%
16	Gaussian	91.85%	85.61%	93.98%	88.27%
32	Gaussian	92.22%	85.69%	94.56%	87.51%
64	Gaussian	91.39%	83.58%	93.85%	85.43%
1	GMM	91.60%	85.01%	91.64%	83.93%
2	GMM	91.07%	82.70%	92.20%	85.60%
4	GMM	91.67%	83.70%	92.64%	84.35%
8	GMM	91.85%	85.46%	93.13%	86.07%
16	GMM	92.10%	85.84%	94.08%	88.59%
32	GMM	92.57%	86.37%	94.69%	87.90%
64	GMM	92.03%	84.06%	94.16%	86.19%

behavior is the randomness involved in training a neural network. Overall, the modifications still seem to improve the performance. When using the modified AdaCos loss and no mixup, the loss diverges because $\hat{s}^{(t)}$ grows exponentially. A proof can be found in the appendix.

E. Using sub-clusters for AdaCos

Now, further experiments are conducted to show that using sub-clusters inside the AdaCos loss improves the outlier detection performance. For this purpose, the neural network is trained with an increasing number of sub-clusters and evaluated with a single Gaussian and a GMM with modes equal to the number of sub-clusters as backends. The corresponding AUCs and pAUCs obtained on the development and evaluation set are shown in Tab. IV.

One can draw two main conclusions from the experimental results. First, using sub-clusters is highly beneficial, especially to increase the performance on the evaluation set. Note that by doing so, it is also possible to exclude potential outliers from the training data by removing small sub-clusters as done in [22]. Thus, this procedure is also well-suited for truly unsupervised anomaly detection problems instead of semi-supervised ones. A second conclusion is that using a GMM instead of a single Gaussian always improves the results because it can be fitted more accurately to the individual sub-clusters of the distribution.

TABLE V

MEAN AUCs AND PAUCs PER MACHINE TYPE OBTAINED WITH DIFFERENT REPRESENTATIONS ON THE DEVELOPMENT AND EVALUATION SET. WHEN USING THE COMBINED REPRESENTATIONS, ONLY THE LEARNED REPRESENTATIONS ARE USED, EXCEPT FOR MACHINE TYPE TOYCONVEYOR WHERE THE MEAN IS USED INSTEAD.

representation	machine type	development set		evaluation set	
		AUC	pAUC	AUC	pAUC
mean	fan	80.73%	66.16%	95.32%	80.62%
max	fan	64.59%	51.48%	78.98%	57.70%
learned	fan	87.61%	77.93%	97.60%	93.24%
mean	pump	82.99%	68.50%	88.24%	70.36%
max	pump	70.13%	59.17%	68.96%	55.08%
learned	pump	94.71%	88.91%	96.76%	88.30%
mean	slider	87.46%	63.95%	72.16%	53.08%
max	slider	93.69%	76.69%	90.55%	70.65%
learned	slider	99.55%	97.63%	97.61%	89.46%
mean	valve	55.59%	50.23%	54.86%	52.09%
max	valve	98.54%	93.08%	96.35%	88.18%
learned	valve	98.63%	94.62%	98.81%	95.80%
mean	ToyCar	94.10%	80.94%	91.54%	76.87%
max	ToyCar	68.36%	53.85%	70.31%	54.90%
learned	ToyCar	96.37%	91.64%	95.99%	91.93%
mean	ToyConveyor	85.78%	67.76%	91.74%	78.13%
max	ToyConveyor	57.51%	50.39%	65.40%	53.06%
learned	ToyConveyor	73.89%	61.22%	81.37%	68.64%
mean	all	80.91%	66.19%	82.31%	68.52%
max	all	76.25%	64.71%	78.42%	63.26%
learned	all	92.57%	86.37%	94.69%	87.90%
combined	all	94.21%	87.13%	96.42%	89.24%

F. Utilizing simple representations

Instead of learning representations of the data by training a neural network, one can also use simple representations directly derived from the data. These representations have the advantage that extracting them does not require any training and they possibly contain useful information about the data that is not needed to discriminate among the classes and thus not contained in the trained representations. On the other hand, it is by no means ensured that these representations contain any useful information for detecting anomalous data at all. In this work, the mean and maximum of the log-Mel frequency bins over time are investigated as alternative representations. For this purpose, their performance is evaluated by estimating their distribution with a single Gaussian component each and using the resulting log-probabilities to detect outliers. The results can be found in Tab. V.

First, it can be seen that for most machine types, except “ToyConveyor”, the trained representations perform best. But for some cases, the simple representations work surprisingly well. Examples are the maximum values for machine type “valve” and mean value for “ToyConveyor”, which even outperforms the learned representation. The reason is that the performance of the learned representation is much worse for “ToyConveyor” than for all other machine types. This behavior can be found for many of the submitted systems e.g. [12], [15] and the exact reasons are still unclear. Koizumi et al. [4] speculate that the normal samples belonging to different machine ids of “ToyConveyor” are very similar and thus

discriminative approaches have difficulties in finding decision boundaries. The other way around, i.e. using the mean value for “valve” and the maximum values for “ToyConveyor” leads to very poor performance, close to random guessing, in both cases. This is the reason why one cannot simply concatenate all three representations to obtain a single representation for all machine types because it would significantly degrade the performance. The best overall performance is achieved, by using the mean representations for “ToyConveyor” and the learned representations for the other machine types.

G. Comparing the performance to other published systems

Last but not least, the presented approach is compared to the five highest-ranked systems submitted to task 2 of the DCASE challenge 2020. To our best knowledge, no more recent work has been published and thus these systems represent the state-of-the-art. The results can be found in Fig. 2 and Fig. 3. First and foremost, the presented system significantly outperforms every other published system, both in terms of AUC and pAUC. This can easily be seen by comparing its performance to the one of the winning system of the challenge [34]. The proposed approach has a higher score for every machine type, except “slider” where the performance is the same. Furthermore, for most systems there is at least one machine type where the performance drops significantly, compared to the other systems, whereas the presented approach performs reasonably well for all machine types.

Note, that all systems but the one submitted by Primus [36] consist of an ensemble of multiple, very different models whereas the presented system consists of just a single model. Thus, for completeness an ensemble consisting of multiple versions of the proposed approach, each trained with another number of sub-clusters, ranging from 2^0 to 2^6 , is also included. The ensemble is realized by summing the log-probabilities of all GMMs belonging to the subsystems. As expected, this ensemble significantly outperforms the system based on a single model and reaches a mean AUC of 97% and a mean pAUC of 91.24%.

V. CONCLUSIONS AND FUTURE WORK

In this work, multiple changes to the standard AdaCos loss specifically aimed at learning lower-dimensional representations for anomalous sound detection are proposed. In experiments conducted on the DCASE 2020 dataset for “Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring”, it is shown that Gaussians or more generally GMMs outperform other widely used backends, namely the direct output of the trained model, cosine similarity and PLDA. Furthermore, using mixup in combination with modified parameter computations for AdaCos further improves the obtained results. By also using multiple learned sub-clusters instead of a single one for each class, less restrictive distributions than a single Gaussian for the representations of the data are learned. As a result, an even higher anomalous sound detection performance is achieved. In an additional experiment, the learned representation is compared to simple

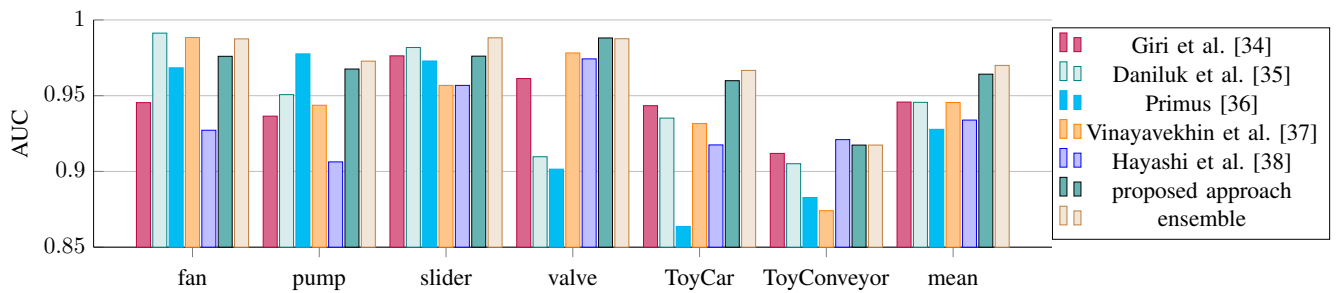


Fig. 2. Comparison of the AUCs obtained on the evaluation set with the top five highest-ranked systems submitted to the DCASE 2020 challenge task 2, the proposed approach and an ensemble. The ensemble consists of the sum of all log-probabilities given by GMMs belonging to trained models of the proposed approach with a different number of sub-clusters, ranging from 2^0 to 2^6 .

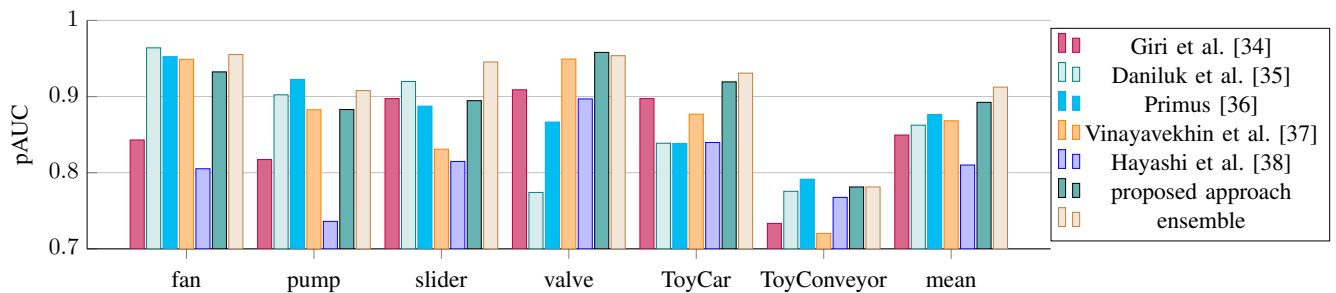


Fig. 3. Comparison of the pAUCs obtained on the evaluation set with the top five highest-ranked systems submitted to the DCASE 2020 challenge task 2, the proposed approach and an ensemble. The ensemble consists of the sum of all log-probabilities given by GMMs belonging to trained models of the proposed approach with a different number of sub-clusters, ranging from 2^0 to 2^6 .

representations, namely the temporal mean and maximum of the log-Mel spectrograms, and is shown to outperform them except for the machine type “ToyConveyor” where using the mean leads to the best results. Last but not least, the presented approach is shown to significantly outperform all other published systems on this dataset even when not ensembling multiple subsystems.

There are still some open questions to be answered. For the machine type “ToyConveyor”, the performance of the learned representations is worse than simply taking the temporal mean of the log-Mel spectrogram. This shows that there is still room for improving the training process of the learned representations. One way to accomplish this could be using a self-supervised learning paradigm instead of training discriminatively among the known classes as done in [14]. Additionally, future work should also be aimed at clarifying why the performance for this particular class is worse to gain insights that may also be helpful for anomalous sound detection in general. Another way to further improve the performance of the presented model is to use more sophisticated approaches to mix samples than plain mixup. A collection of ways to mix samples can be found in [39].

REFERENCES

- [1] Pasquale Foggia, Nicolai Petkov, Alessia Saggese, Nicola Strisciuglio, and Mario Vento, “Audio surveillance of roads: A system for detecting anomalous sounds,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 279–288, 2016.
- [2] Yanxiong Li, Xianku Li, Yuhan Zhang, Mingle Liu, and Wucheng Wang, “Anomalous sound detection using deep audio representation and a BLSTM network for audio surveillance of roads,” *IEEE Access*, vol. 6, pp. 58043–58055, 2018.
- [3] Tomoki Hayashi, Tatsuya Komatsu, Reishi Kondo, Tomoki Toda, and Kazuya Takeda, “Anomalous sound event detection based on wavenet,” in *26th European Signal Processing Conference (EUSIPCO)*. 2018, pp. 2494–2498, IEEE.
- [4] Yuma Koizumi, Yohei Kawaguchi, Keisuke Imoto, Toshiaki Nakamura, Yuki Nikaïdo, Ryo Tanabe, Harsh Purohit, Kaori Suefusa, Takashi Endo, Masahiro Yasuda, and Noboru Harada, “Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring,” in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 81–85.
- [5] Charu Aggarwal, *Outlier Analysis*, Springer, 2nd edition, 2017.
- [6] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, “ArcFace: Additive angular margin loss for deep face recognition,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4690–4699, IEEE.
- [7] Xiao Zhang, Rui Zhao, Yu Qiao, Xiaogang Wang, and Hongsheng Li, “AdaCos: Adaptively scaling cosine logits for effectively learning deep face representations,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 10823–10832, IEEE.
- [8] Stawomir Kapka, “ID-conditioned auto-encoder for unsupervised anomaly detection,” in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 71–75.
- [9] Koichi Miyazaki, Tatsuya Komatsu, Tomoki Hayashi, Shinji Watanabe, Tomoki Toda, and Kazuya Takeda, “Conformer-based sound event detection with semi-supervised learning and data augmentation,” in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 100–104.
- [10] Tadanobu Inoue, Phongtharin Vinayavekhin, Shu Morikuni, Shiqiang Wang, Tuan Hoang Trong, David Wood, Michiaki Tatsubori, and Ryuki Tachibana, “Detection of anomalous sounds for machine condition monitoring using classification confidence,” in *Detection and Classification*

of *Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 66–70.

[11] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao, “A discriminative feature learning approach for deep face recognition,” in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 499–515.

[12] Jose A. Lopez, Hong Lu, Paulo Lopez-Meyer, Lama Nachman, Georg Stemmer, and Jonathan Huang, “A speaker recognition approach to anomaly detection,” in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 96–99.

[13] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu, “Additive margin softmax for face verification,” *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[14] Ritwik Giri, Srikanth V. Tenneti, Fangzhou Cheng, Karim Helwani, Umut Isik, and Arvindh Krishnaswamy, “Self-supervised classification for detecting anomalous sounds,” in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 46–50.

[15] Qiping Zhou, “ArcFace based sound mobilenets for DCASE 2020 task 2,” Tech. Rep., DCASE2020 Challenge, 2020.

[16] Simon J.D. Prince and James H. Elder, “Probabilistic linear discriminant analysis for inferences about identity,” in *11th International Conference on Computer Vision. ICCV. IEEE*, 2007, pp. 1–8.

[17] Kevin Wilkinghoff, “Using look, listen, and learn embeddings for detecting anomalous sounds in machine condition monitoring,” in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 215–219.

[18] Kilian Q. Weinberger and Lawrence K. Saul, “Distance metric learning for large margin nearest neighbor classification,” *Journal of machine learning research*, vol. 10, no. 2, 2009.

[19] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz, “Mixup: Beyond empirical risk minimization,” in *International Conference on Learning Representations (ICLR)*, 2018.

[20] Manli Zhu and Aleix M. Martinez, “Subclass discriminant analysis,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 8, pp. 1274–1286, 2006.

[21] Kateryna Chumachenko, Alexandros Iosifidis, and Moncef Gabbouj, “Robust fast subclass discriminant analysis,” in *28th European Signal Processing Conference (EUSIPCO)*. IEEE, 2020, pp. 1397–1401.

[22] Jiankang Deng, Jia Guo, Tongliang Liu, Mingming Gong, and Stefanos Zafeiriou, “Sub-center ArcFace: Boosting face recognition by large-scale noisy web faces,” in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 741–757.

[23] Harsh Purohit, Ryo Tanabe, Takeshi Ichige, Takashi Endo, Yuki Nikaido, Kaori Suefusa, and Yohei Kawaguchi, “MIMII Dataset: Sound dataset for malfunctioning industrial machine investigation and inspection,” in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. 2019, pp. 209–213, New York University.

[24] Yuma Koizumi, Shoichiro Saito, Hisashi Uematsu, Noboru Harada, and Keisuke Imoto, “ToyADMOS: A dataset of miniature-machine operating sounds for anomalous sound detection,” in *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 313–317.

[25] Jason Cramer, Ho-Hsiang Wu, Justin Salamon, and Juan Pablo Bello, “Look, listen, and learn more: Design choices for deep audio embeddings,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3852–3856.

[26] Sascha Grollmisch, David Johnson, Jakob Abeßer, and Hanna Lukashevich, “IAEO3 - combining OpenL3 embeddings and interpolation autoencoder for anomalous sound detection,” Tech. Rep., DCASE2020 Challenge, 2020.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778, IEEE.

[28] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *30th International Conference on Machine Learning (ICML)*, 2013.

[29] Sergey Ioffe and Christian Szegedy, “Batch normalization: accelerating deep network training by reducing internal covariate shift,” in *32nd International Conference on Machine Learning (ICML)*, 2015, vol. 37, pp. 448–456.

[30] Diederik P. Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations (ICLR)*, Yoshua Bengio and Yann LeCun, Eds., 2015.

[31] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., “Tensorflow: A system for large-scale machine

learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[32] Aleksandr Sizov, Kong Aik Lee, and Tomi Kinnunen, “Unifying probabilistic linear discriminant analysis variants in biometric authentication,” in *Proc. S+SSPR*. Springer, 2014, pp. 464–475, Software available at <https://sites.google.com/site/fastplda/>.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] Ritwik Giri, Srikanth V. Tenneti, Karim Helwani, Fangzhou Cheng, Umut Isik, and Arvindh Krishnaswamy, “Unsupervised anomalous sound detection using self-supervised classification and group masked autoencoder for density estimation,” Tech. Rep., DCASE2020 Challenge, 2020.

[35] Pawel Daniluk, Marcin Gozdziowski, Slawomir Kapka, and Michal Kosmider, “Ensemble of auto-encoder based systems for anomaly detection,” Tech. Rep., DCASE2020 Challenge, 2020.

[36] Paul Primus, “Reframing unsupervised machine condition monitoring as a supervised classification task with outlier-exposed classifiers,” Tech. Rep., DCASE2020 Challenge, 2020.

[37] Phongtharin Vinayavekhin, Tadanobu Inoue, Shu Morikuni, Shiqiang Wang, Tuan Hoang Trong, David Wood, Michiaki Tatsubori, and Ryuki Tachibana, “Detection of anomalous sounds for machine condition monitoring using classification confidence,” Tech. Rep., DCASE2020 Challenge, 2020.

[38] Tomoki Hayashi, Takenori Yoshimura, and Yusuke Adachi, “Conformer-based id-aware autoencoder for unsupervised anomalous sound detection,” Tech. Rep., DCASE2020 Challenge, 2020.

[39] Cecilia Summers and Michael J. Dinneen, “Improved mixed-example data augmentation,” in *Winter Conference on Applications of Computer Vision (WACV)*. 2019, pp. 1262–1270, IEEE.

APPENDIX

Remark. *The adaptive scale parameter $\hat{s}^{(t)}$ of the modified AdaCos loss grows exponentially when not using mixup.*

Proof. After a few training iterations without mixup i.e. for $t > t_0 \in \mathbb{N}$, most training samples will have a very small angle to their associated class, i.e. $\theta_{i,y_i} \approx 0$. Therefore, $\cos \theta_{i,y_i} \approx 1$ and thus also $\cos \hat{\theta}_{\text{med}}^{(t)} \approx 1$. Furthermore, as empirically shown in [7], on average $\theta_{i,k} < \frac{\pi}{2}$ and thus $\cos \theta_{i,k} > 0$ for most $k \neq y_i$. Hence, by using the fact that the logarithm is a concave function and applying Jensen’s inequality we obtain

$$\begin{aligned}
 \hat{s}^{(t)} &= \frac{f_{\max}^{(t)} + \log \hat{B}_{\text{avg}}^{(t)}}{\cos \left(\min \left(\frac{\pi}{4}, \hat{\theta}_{\text{med}}^{(t)} \right) \right)} \\
 &\approx \log \left(\frac{1}{N} \sum_{i \in \mathcal{N}^{(t)}} \sum_{k=1}^{CS} \exp \left(\hat{s}^{(t-1)} \cdot \cos \theta_{i,k} \right) \right) \\
 &\geq \frac{1}{N} \sum_{i \in \mathcal{N}^{(t)}} \sum_{k=1}^{CS} \hat{s}^{(t-1)} \cdot \cos \theta_{i,k} \\
 &\approx \hat{s}^{(t-1)} \underbrace{\left(1 + \frac{1}{N} \sum_{i \in \mathcal{N}^{(t)}} \sum_{\substack{k=1 \\ k \neq y_i}}^{CS} \cos \theta_{i,k} \right)}_{>0}
 \end{aligned} \tag{10}$$

showing that $\hat{s}^{(t)}$ grows exponentially when not using mixup. Note that this inequality does not hold if only mixed-up samples are used for training. The reason is that most samples belong to multiple classes and thus do not have an angle of approximately 0 to their corresponding class mean because AdaCos increases the margin between classes.

A.1.2 *Key publication 2*

Kevin Wilkinghoff, Alessia Cornaggia-Urrigshardt, and Fahrettin Gökgöz. “Two-Dimensional Embeddings for Low-Resource Keyword Spotting Based on Dynamic Time Warping.” In: *14th ITG Conference on Speech Communication*. VDE-Verlag, 2021, pp. 9–13.

© 2021 VDE, published with IEEE.

The co-authors of this publication contributed in the following ways: *Alessia Cornaggia-Urrigshardt* applied voice activity detection ([VAD](#)) as pre-processing and [DTW](#) to the resulting embeddings. She also wrote Sections 2.3, 2.4, 2.5 and 3.3, and created Figures 1 and 2. *Fahrettin Gökgöz* collected and prepared the dataset used for the experiments.

Two-Dimensional Embeddings for Low-Resource Keyword Spotting Based on Dynamic Time Warping

Kevin Wilkinghoff¹, Alessia Cornaggia-Urrigshardt¹, Fahrettin Gökgöz

Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE, Wachtberg, Germany
Email: {kevin.wilkinghoff, alessia.cornaggia-urrigshardt, fahrettin.goekgoez}@fkie.fraunhofer.de

Abstract

State-of-the-art keyword spotting systems consist of neural networks trained as classifiers or trained to extract discriminative representations, so-called embeddings. However, a sufficient amount of labeled data is needed to train such a system. Dynamic time warping is another keyword spotting approach that uses only a single sample of each keyword as patterns to be searched and thus does not require any training. In this work, we propose to combine the strengths of both keyword spotting approaches in two ways: First, an angular margin loss for training a neural network to extract two-dimensional embeddings is presented. It is shown that these embeddings can be used as features for dynamic time warping, outperforming cepstral features even when very few training samples are available. Second, dynamic time warping is applied to cepstral features to turn weak into strong labels and thus provide more labeled training data for the two-dimensional embeddings.

1 Introduction

The goal of keyword spotting (KWS) is to detect specific keywords from a small application-dependent set of words in audio recordings. Therefore, large parts of the audio recordings are not of interest because they do not contain these keywords. In contrast to large-vocabulary continuous speech recognition systems that transcribe all words being spoken and thus are also able to detect a specified subset of keywords, keyword spotting systems require less labeled training data and less computational power. This is advantageous for speech recognition applications with a restricted vocabulary where only limited resources in terms of data and computational power are available. Examples are rarely spoken languages for which only insufficient labeled data is available [1], and difficult (acoustic) environments where labeled data recorded under the same acoustic conditions and also computational power is restricted [2].

An unsupervised approach to detect keywords in audio recordings is to use dynamic time warping (DTW) [3, 4]. Its fundamental idea is to search for a set of provided keywords represented as features that do not require any training in an audio file. This is called query-by-example KWS and has the advantage that no training is needed and it thus works with very limited amounts of data. When at least some labeled data is available, many KWS systems are trained to extract discriminative representations, so-called embeddings, from audio recordings. In [5], it has been shown that supervised embeddings extracted with linear discriminant analysis and graph embeddings outperform unsupervised embeddings and DTW-based KWS systems. Other works utilize different types of neural networks to extract embeddings, for example (Siamese) convolutional neural networks (CNNs) [6], CNNs trained by minimiz-

ing an additive angular margin loss [7] or long-short term memory networks (LSTMs) [8, 9]. Using a simple sliding window in combination with these discriminative embeddings has been shown to outperform unsupervised DTW-based approaches. However, most of these systems, especially LSTM-based ones, require much labeled training data in order to work well. In [9], 1115 hours of speech and in [8] 2500 hours of speech are used for training the LSTM-based embeddings. To circumvent this problem, Menon et al. used only 34 minutes of labeled speech to generate labels with a DTW-based approach for a larger unlabeled dataset, which then was used to train a CNN [1]. Another drawback of using supervised embeddings is that a fixed-sized sliding window is less accurate than DTW when predicting the exact on- and offset of detections. This is especially problematic for applications where sequences of keywords and their order need to be recognized. Examples are telephone numbers, PINs or more generally passwords or -phrases, coordinates or spelled words.

The goal of this work is to combine the strengths of both KWS approaches. To this end, an angular margin loss for learning two-dimensional embeddings and a DTW-based system utilizing these embeddings for spotting sequences of keywords are proposed. The presented system is evaluated on an internal dataset recorded at Fraunhofer FKIE consisting of spoken coordinates in German and a very small training set of less than 7 minutes. It is shown that the presented two-dimensional embeddings outperform cepstral features, more precisely human factor cepstral coefficients (HFCC-ENS) [4]. In additional experiments, cepstrum-based DTW is used to automatically convert weak into strong labels by predicting on- and offset of keywords that are known to be present in a recording. When providing these automatically labeled data as additional training data, the performance obtained with the two-dimensional embeddings is significantly improved.

2 Methodology

2.1 Two-dimensional AdaCos loss

Embeddings obtained by minimizing an angular margin loss function such as ArcFace [10] have been shown to outperform other embeddings and yield state-of-the-art classification performances. The angular margin loss AdaCos [11] does not depend on any hyperparameters to be tuned but uses an adaptive scale parameter while performing equally well as ArcFace. Thus, we propose to modify AdaCos to learn two-dimensional embeddings suitable for being used with DTW.

Let $x_i \in \mathbb{R}^{T_i \times D}$ be an embedding belonging to keyword k of the $K \in \mathbb{N}$ keywords with time dimension T_i and feature dimension D . In standard one-dimensional AdaCos, mean values are learned for each class such that the cosine similarities of all samples to their corresponding class

¹These authors contributed equally.

means are maximized while the cosine similarities to other class means are minimized. Moreover, a margin between the classes is ensured. For two-dimensional AdaCos, a set of $T \in \mathbb{N}$ embeddings $\mathcal{E}_k \subset \mathbb{R}^D$ instead of a single mean embedding is learned for each keyword. Then, the cosine angle $\theta_{i,k} \in [0, \pi]$ between x_i and \mathcal{E}_k is defined through

$$\cos \theta_{i,k} = \frac{1}{T_i} \sum_{t=1}^{T_i} \max_{e_j \in \mathcal{E}_k} \frac{\langle x_{i,t}, e_j \rangle}{\|x_{i,t}\|_2 \|e_j\|_2}. \quad (1)$$

This definition of the angle is the only difference of the two-dimensional AdaCos to the regular one-dimensional loss. Therefore, the probability of sample x_i belonging to keyword j is still given by

$$P_{i,j} := \frac{\exp(\tilde{s} \cdot \cos \theta_{i,j})}{\sum_{k=1}^K \exp(\tilde{s} \cdot \cos \theta_{i,k})} \quad (2)$$

with the standard adaptive scale parameter \tilde{s} (see [11]).

To ensure that all audio samples of keywords used to train the neural network are of length 0.5 seconds, shorter samples are zero-padded and a sliding window of size 0.5 seconds and a step size of 0.1 seconds is used for longer samples. To obtain two-dimensional embeddings for the test sentences, which are much longer than 0.5 seconds, first a sliding window of size 0.5 seconds with a hop size of $1/T$ is used to compute another embedding for each window position. Then, all embeddings belonging to a single test recording are combined into one longer embedding by taking the mean of all $1 \times D$ -dimensional vectors of the embeddings for which their corresponding windows overlap at a given position.

To obtain a database consisting of a single representation for each keyword, the DTW barycenter averaging (DBA) algorithm [12] as implemented in [13] is used. Instead of computing a regular Euclidean mean of all samples belonging to a single keyword, which minimizes the Euclidean distance, this algorithm estimates the more general Fréchet mean that minimizes the distance implied by DTW. As a result, the examples stored in the keyword database are more similar to corresponding keyword samples when searching for a keyword with DTW and thus the performance is improved.

2.2 Network architecture

The architecture of the neural network used for extracting two-dimensional embeddings is shown in Tab. 1. It consists of a modified ResNet architecture [14] in combination with the 2D AdaCos loss and, except for the loss function, strongly resembles the architecture used in [15]. Each residual block includes *batch normalization* layers [16] and *LeakyReLU* [17] with $\alpha = 0.1$ as nonlinearities. As input to the neural network, *log-Mel spectrograms* with 64 Mel-bins, a window size of 1024 and a hop size of 256 are extracted from raw waveforms of length 0.5 seconds, resulting in features of size 32×64 . To avoid overfitting of the model to the limited amount of training data, two data augmentation techniques are used: 1) *SpecAugment* [18], which consists of frequency masking, time masking and time warping, and 2) *mixup* [19], more precisely *manifold mixup* [20], for random linear interpolations between hidden representations of training data. To take the usage of mixup into account when minimizing the two-dimensional AdaCos loss while training, the extension presented in [15] is used. In all experiments, the network, implemented in

layer name	structure	output size
input	-	32×64
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1×1	$32 \times 64 \times 16$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1×2	$32 \times 32 \times 32$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1×2	$32 \times 16 \times 64$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1×2	$32 \times 8 \times 128$
max pooling	1×8 , stride= 1×1	32×128
dense (representation)	linear	32×32
2D AdaCos	-	38

Table 1: Modified ResNet architecture used for extracting two-dimensional embeddings.

Tensorflow [21], is trained for 1000 epochs with a batch size of 32 using *Adam* [22] for optimization.

2.3 Voice activity detection (VAD)

A pre-processing step for analyzing long audio recordings or a post-processing step to refine the results of the proposed KWS system (cf. Sec. 2.4) is a *Voice Activity Detection* (VAD) algorithm. We propose a VAD based on a combination of *spectral energy* and two particular audio features, namely *spectral flux* and *spectral flatness*. The use of energy is restricted to pre-defined frequency sub-bands, one for low frequencies in the range of 100-1000 Hz focusing on the tonal components of speech and neglecting low-frequency noise, and one for higher frequencies in the range of 5500-8000 Hz to consider e.g. sibilants. In addition, two audio features, which have been successfully used for speech detection tasks ([23–25]), are used to capture the particular characteristics of speech signals. These features are combined by thresholding the corresponding feature curves. Thresholds were fixed empirically by evaluating the training data. The performance of the VAD evaluated on manually labeled training data has a TP-rate of 99.46% and a FP-rate of 0.77%.

2.4 DTW-based keyword spotting (KWS)

In the proposed query-by-example approach, every *keyword* and target sentence is represented by a sequence of feature vectors, a 2D-feature matrix, obtained from the previously derived *embeddings*, allowing a frame-wise comparison of the target signal with the trained keyword patterns. Let $\mathbb{K}_i \in \mathcal{K}$ be the feature representation of keyword i from the keyword set \mathcal{K} and $\mathbb{S} \in \mathcal{S}$ be the corresponding feature representation of a target signal. To account for the different lengths of keyword utterances \mathbb{K}_i and the sequences \mathbb{S} , we apply sub-sequence DTW [26, 27] to align keywords with sub-sequences of the target signal. In classical DTW approaches, two sequences of feature vectors are time-aligned by calculating a pair-wise similarity between all the feature vectors, thus obtaining a cost matrix \mathbb{C} , which is transformed into an accumulated cost matrix \mathbb{D} following pre-defined step size conditions. Sub-sequence DTW applies the same technique but returns several possible matchings by extracting local maxima of the resulting accumulated cost matrix. As opposed to *diagonal matching*, the step size condition of classical DTW is extended to larger steps, in our case to $\{(2, 1), (1, 1), (1, 2)\}$, to account for faster and slower speakers. \mathbb{C} is calculated using the inner product of any two feature vectors j and k ,

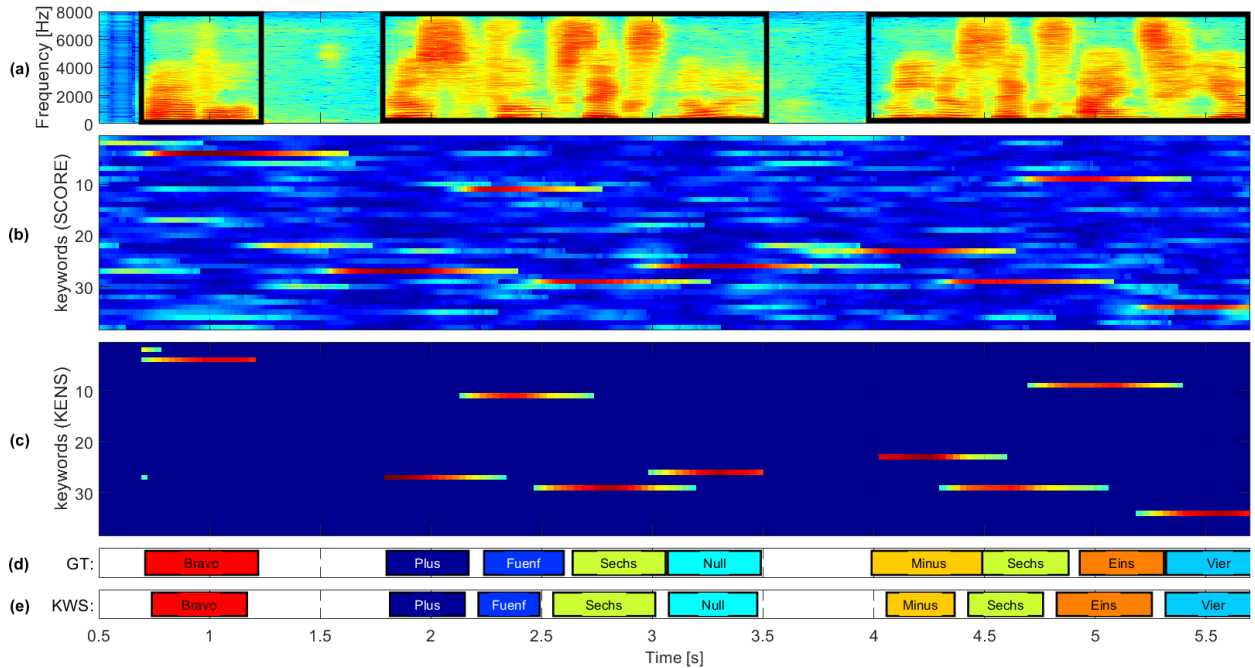


Figure 1: DTW-KENS-KWS: *Dynamic Time Warping Keyword Energy Normalized Statistics Keyword Spotting* showing (a) the spectrogram of a target signal (with VAD results indicated by black boxes), (b) the score matrix S_{DTW} , (c) the enhanced KENS matrix, (d) a manually generated ground truth annotation, and (e) the keyword spotting results.

i.e. $Sim^{K_i, S}(j, k) := \langle v_j^{K_i}, v_k^S \rangle / \|v_j^{K_i}\|_2 \|v_k^S\|_2$, where $v_j^{K_i}$ denotes the j^{th} feature vector of a keyword K_i and v_k^S the k^{th} the one of a target sentence (cf. [4]). Then $C = 1 - Sim^{K_i, S}$.

2.5 DTW-KENS-KWS

Instead of extracting paths on the basis of an accumulated cost matrix for each keyword K_i , we only consider the last row of D , which can be interpreted as a cost curve and, by reverting the negation, as a *score* function s_{DTW}^i for a keyword K_i . All s_{DTW}^i are concatenated to a *score matrix* S_{DTW} for all $K_i \in \mathcal{K}$ and a given signal $S \in \mathcal{S}$, where high values indicate likely matchings. An example of S_{DTW} is given in Fig. 1 (b), where red colors indicate a high score. The y-axis shows the index of the keywords (cf. 3.1).

To enhance the high score regions, we apply a technique from music processing used to calculate CENS features [28, 29], which has also been adopted for keyword spotting [4] (cf. Sec. 3.2), consisting of *normalization* and *quantization*, *smoothing*, and *downsampling*. These operations are applied to the DTW-based score matrix representing *keywords*. Hence we refer to the resulting sequence of post-processed feature vectors by *Keyword Energy Normalized Statistics* (KENS). An example of this KENS matrix is shown in Fig. 1 (c).

Keyword spotting is performed by picking the column-wise maxima of the KENS matrix and extracting subblocks satisfying predefined lengths. Larger blocks are divided into consecutive matches of the same keyword by considering personalized average lengths. In addition, the proposed VAD (Sec. 2.3) is used to adjust the start and end times of the detected keywords. The VAD results are indicated by the black boxes in Fig. 1 (a). Furthermore, the results can be post-processed by using knowledge about the data such as in the given case of spoken coordinates where letters only appear at the beginning of blocks of numbers.

3 Experimental Results

3.1 Dataset

The dataset used for all evaluations contained in this paper is an internal dataset recorded by Fraunhofer FKIE. The corpus consists of read coordinates in the German language. All recordings are downsampled to 16kHz and the annotations have been manually verified twice. The training subset of the dataset consists of 37 keywords and an additional class *Silence* consisting of all six background noise files of Google Speech Commands [30]. The keywords are: *Acht, Alpha, Bravo, Charlie, Delta, Drei, Echo, Eins, Foxtrot, Fünf, Fünnef, Golf, Hotel, Ich berichtige, India, Juliett, Kilo, Korrektur, Korrigiere, Lima, Mike, Minus, Neun, November, Null, Plus, Quebec, Sechs, Sieben, Sierra, Tango, Victor, Vier, Whiskey, X-ray, Yankee, Zwo*. These keywords are all read at most once by 12 male and 7 female speakers, 3 of which are non-native. The recordings were done in different office rooms using standard sound pressure microphones. As a result, there are 16 to 19 versions of each keyword for training with a total duration of less than 7 minutes. For validation and testing, the same speakers read up to 201 sentences each, resulting in a validation set with a duration of 179 minutes containing 1723 sentences and a test set with a duration of 240 minutes containing 2090 sentences.

3.2 Comparison of features

As a first experiment, it is shown that the proposed two-dimensional embeddings outperform classical cepstral features. HFCC-ENS features have been shown to outperform Mel-frequency cepstral coefficients (MFCCs) when detecting keywords with DTW [4]. Therefore, we only consider the former. HFCC-ENS are computed by applying particular filterbanks optimized for human audio perception to spectral feature vectors and then post-processed by nor-

feature	loss	WER	
		validation set	test set
HFCC-ENS	–	0.2533	0.2660
embedding	flatten + softmax	0.3118	0.3115
embedding	2D AdaCos	0.2213	0.2100

Table 2: Word error rates obtained with different features.

malization, quantization, smoothing, and downsampling – a technique applied also in our proposed KWS algorithm (Sec. 2.5). In our evaluation, we use these features together with sub-sequence DTW to extract keywords. The results are post-processed by exploiting domain knowledge, in particular using keyword lengths and allowing letters only at the beginning of number clusters. As opposed to the KENS *embeddings* approach, several examples of the same personalized keyword are used for the DTW KWS method.

The resulting WERs can be found in Tab. 2. Recall that the training dataset is very small and only consists of less than 7 minutes of speech. Still, the two-dimensional embeddings extracted with the proposed 2D AdaCos result in significantly lower WERs than HFCC-ENS features. To verify that the loss is actually important, the network architecture was altered by replacing the 2D AdaCos loss with a flattening operation in combination with a softmax loss. This led to a much higher WER, even higher than the one obtained with HFCC-ENS features.

3.3 Turning weak into strong labels

When using data-driven models such as neural networks, their true power comes from using as much high-quality training data as possible. Creating weak labels for an unlabeled dataset is much less time-consuming than strong labeling because experts can simply listen once and write down all words while listening instead of stopping for each word and marking the precise start- and endpoints. However, weakly labeled data cannot be used to train the presented neural network because it is only known what is being said in a sentence but not when the actual words begin or end. Hence, we propose to use DTW with cepstral features to automatically convert weak into strong labels and thus create additional training data. Since prior knowledge in the form of weak labels is available, this procedure should lead to training data of higher quality than when creating strong labels from scratch as done in [1].

To this end, all the keywords \mathcal{K}_i , given by a fixed feature representation $\mathcal{F}(\mathcal{K}_i)$ – in our case HFCC-ENS, are concatenated in the order they appear in a given training sentence \mathcal{S} , also transformed into the corresponding feature representation $\mathcal{F}(\mathcal{S})$, resulting in a super-feature matrix $\mathcal{F}(\mathcal{K})$. This matrix is then aligned with the sequence of feature vectors of \mathcal{S} , providing a global start and end time of the best alignment of the super-keyword \mathcal{K} . Knowing the length of each *sub*-keyword \mathcal{K}_i allows to sub-divide the matching result. In order to evaluate the performance of the automatic segmentation, which is shown in Fig. 2, we consider the overlap between each resulting segment and its ground truth. This overlap can be interpreted as a percentage of the result interval (x -axis) and of the ground truth (y -axis). The colors indicate the percentage of segments which are considered correct (in %/100). One example is the white point on Fig. 2: 90% of the segments can be seen as correct, if we require the overlap of a ground truth segment with the corresponding result segment to be

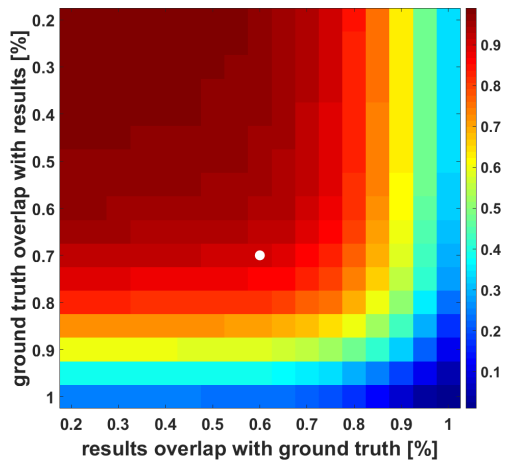


Figure 2: Performance of the automatic segmentation.

automatically labeled data used for:		WER
neural network	Fréchet mean	
		0.2100
	✗	0.1841
✗		0.0724
✗	✗	0.0720

Table 3: Word error rates obtained on the test set using automatically labeled data as additional training data.

at least 70% of the length of the ground truth and 60% of the length of the result segment – which for the purpose of data augmentation may be sufficient. As a result, the total length of the training dataset is increased from less than 7 minutes to 95 minutes.

The effects of using automatically labeled data as additional training data can be found in Tab. 3. As expected, the WER decreases significantly when using more data for training. This is especially true for the neural network, where the WER is reduced by a factor of 3. Hence, the two-dimensional embeddings improve in representing the keywords as features when more training data is available.

4 Conclusions and Future Work

In this work, a keyword spotting system capable of detecting sequences of keywords in low-resource settings has been presented. The system is based on two-dimensional embeddings obtained by training a neural network with a novel 2D AdaCos loss and utilizes these embeddings as features for DTW-based keyword spotting. In experiments conducted on a coordinate recognition dataset in German with a training set of less than 7 minutes, it has been shown that the features lead to a lower WER than HFCC-ENS features. Furthermore, a procedure for converting weak into strong labels has been proposed and shown to significantly improve the performance of the two-dimensional embeddings by generating more training data.

For future work, it can be investigated whether using soft-DTW [31] inside the AdaCos layer leads to an improved performance of the two-dimensional embeddings when using DTW. Furthermore, it is planned to conduct experiments where the two-dimensional embeddings are used in combination with prototypical networks [32] for few-shot learning.

References

- [1] R. Menon, H. Kamper, J. Quinn, and T. Niesler, “Fast ASR-free and almost zero-resource keyword spotting using DTW and CNNs for humanitarian monitoring,” in *19th Annual Conference of the International Speech Communication Association (Interspeech)*, pp. 2608–2612, ISCA, 2018.
- [2] H.-C. Schmitz, F. Kurth, K. Wilkinghoff, U. Müller-schkowski, C. Karrasch, and V. Schmid, “Towards robust speech interfaces for the ISS,” in *International Conference on Intelligent User Interfaces (IUI) Companion*, pp. 110–111, ACM, 2020.
- [3] M. D. Wachter, M. Matton, K. Demuynck, P. Wambacq, R. Cools, and D. V. Compennolle, “Template-based continuous speech recognition,” *IEEE Trans. Speech Audio Process.*, vol. 15, no. 4, pp. 1377–1390, 2007.
- [4] D. Von Zeddelmann, F. Kurth, and M. Müller, “Perceptual audio features for unsupervised key-phrase detection,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 257–260, IEEE, 2010.
- [5] K. Levin, K. Henry, A. Jansen, and K. Livescu, “Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings,” in *Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 410–415, IEEE, 2013.
- [6] H. Kamper, W. Wang, and K. Livescu, “Deep convolutional acoustic word embeddings using word-pair side information,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4950–4954, IEEE, 2016.
- [7] H. Ma, Y. Bai, J. Yi, and J. Tao, “Hypersphere embedding and additive margin for query-by-example keyword spotting,” in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 868–872, IEEE, 2019.
- [8] G. Chen, C. Parada, and T. N. Sainath, “Query-by-example keyword spotting using long short-term memory networks,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5236–5240, IEEE, 2015.
- [9] J. Hou, L. Xie, and Z. Fu, “Investigating neural network based query-by-example keyword spotting approach for personalized wake-up word detection in mandarin chinese,” in *10th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pp. 1–5, IEEE, 2016.
- [10] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “ArcFace: Additive angular margin loss for deep face recognition,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4690–4699, IEEE, 2019.
- [11] X. Zhang, R. Zhao, Y. Qiao, X. Wang, and H. Li, “AdaCos: Adaptively scaling cosine logits for effectively learning deep face representations,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10823–10832, IEEE, 2019.
- [12] F. Petitjean, A. Ketterlin, and P. Gançarski, “A global averaging method for dynamic time warping, with applications to clustering,” *Pattern recognition*, vol. 44, no. 3, pp. 678–693, 2011.
- [13] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods, “Tslearn, a machine learning toolkit for time series data,” *Journal of Machine Learning Research*, vol. 21, no. 118, pp. 1–6, 2020.
- [14] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, IEEE, 2016.
- [15] K. Wilkinghoff, “Sub-cluster AdaCos: Learning representations for anomalous sound detection,” in *International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2021.
- [16] S. Ioffe and C. Szegedy, “Batch normalization: accelerating deep network training by reducing internal covariate shift,” in *32nd International Conference on Machine Learning (ICML)*, pp. 448–456, 2015.
- [17] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *30th International Conference on Machine Learning (ICML)*, 2013.
- [18] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “SpecAugment: A simple data augmentation method for automatic speech recognition,” in *20th Annual Conference of the International Speech Communication Association (Interspeech)*, pp. 2613–2617, ISCA, 2019.
- [19] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “Mixup: Beyond empirical risk minimization,” in *International Conference on Learning Representations (ICLR)*, 2018.
- [20] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio, “Manifold mixup: Better representations by interpolating hidden states,” in *36th International Conference on Machine Learning (ICML)*, vol. 97 of *Proceedings of Machine Learning Research*, pp. 6438–6447, PMLR, 2019.
- [21] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, “Tensorflow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 265–283, 2016.
- [22] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations (ICLR)*, 2015.
- [23] Y. Ma and A. Nishihara, “Efficient voice activity detection algorithm using long-term spectral flatness measure,” *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2013, no. 1, pp. 1–18, 2013.
- [24] S. Urrigshardt, S. Kreuzer, and F. Kurth, “General detection of speech signals in the time-frequency plane,” in *ITG Symposium on Speech Communication*, pp. 1–5, VDE, 2016.
- [25] S. Lee, J. Kim, and I. Lee, “Speech/audio signal classification using spectral flux pattern recognition,” in *Workshop on Signal Processing Systems*, pp. 232–236, IEEE, 2012.
- [26] F. Kurth and D. von Zeddelmann, “An analysis of mfcc-like parametric audio features for keyphrase spotting applications,” *ITG Symposium on Speech Communication*, 2010.
- [27] D. von Zeddelmann, F. Kurth, and M. Müller, “Vergleich von Matching-Techniken für die Detektion gesprochener Phrasen,” *Deutsche Jahrestagung für Akustik*, pp. 257–260, 2010.
- [28] M. Müller, F. Kurth, and M. Clausen, “Chroma-based statistical audio features for audio matching,” in *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 275–278, IEEE, 2005.
- [29] M. Müller, F. Kurth, and M. Clausen, “Audio matching via chroma-based statistical features,” in *6th International Conference on Music Information Retrieval (ISMIR)*, pp. 288–295, 2005.
- [30] P. Warden, “Speech commands: A dataset for limited-vocabulary speech recognition,” *CoRR*, vol. abs/1804.03209, 2018.
- [31] M. Cuturi and M. Blondel, “Soft-DTW: a differentiable loss function for time-series,” in *34th International Conference on Machine Learning (ICML)*, pp. 894–903, PMLR, 2017.
- [32] J. Snell, K. Swersky, and R. S. Zemel, “Prototypical networks for few-shot learning,” in *Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 4077–4087, 2017.

A.1.3 *Key publication 3*

Kevin Wilkinghoff and Alessia Cornaggia-Urrigshardt. “On choosing decision thresholds for anomalous sound detection in machine condition monitoring.” In: *24th International Congress on Acoustics*. The Acoustical Society of Korea, 2022. © 2022 The Acoustical Society of Korea.

The co-author of this publication contributed in the following ways: *Alessia Cornaggia-Urrigshardt* assisted with reviewing literature and implemented some of the evaluation methods. She also wrote parts of Section 3.

On choosing decision thresholds for anomalous sound detection in machine condition monitoring

Kevin WILKINGHOFF¹, Alessia CORNAGGIA-URRIGSHARDT²

¹Fraunhofer FKIE, Germany, kevin.wilkinghoff@fkie.fraunhofer.de

²Fraunhofer FKIE, Germany, alessia.cornaggia-urrigshardt@fkie.fraunhofer.de

ABSTRACT

Most anomalous sound detection (ASD) systems output a score for each audio sample presented to the system. Ideally, these anomaly scores differ for normal and anomalous samples such that one can determine whether a given sample is normal or anomalous by comparing the scores to predefined thresholds. However, determining these thresholds is non-trivial, especially when no anomalous samples are provided as training data. In this work, several methods for finding such decision thresholds are evaluated and compared to each other when acoustically monitoring the condition of machines in noisy environments. To this end, the state-of-the-art in ASD for machine condition monitoring will be reviewed first. Using a state-of-the-art ASD system, experimental evaluations are conducted on the DCASE 2020 ASD dataset to evaluate differently attained decision thresholds.

Keywords: anomalous sound detection, decision threshold, machine listening

1 INTRODUCTION

Anomaly detection [1, 2] is the task of identifying samples substantially differing from normal samples that are frequently encountered. Collecting these *anomalous* samples is difficult because by definition anomalies occur only rarely and often are very costly to produce artificially. For example, when acoustically monitoring the condition of machines, creating anomalous samples of machine sounds translates to damaging potentially costly machines in very specific ways whereas recording fully functioning machines is much less costly. Furthermore, for many applications anomalies are very diverse making it practically impossible to sufficiently cover the space of anomalous samples by collecting as many as possible of them. Hence, in many cases anomaly detection takes place in a semi-supervised setting meaning that only normal samples are available for training a system.

Evaluating and comparing different systems for anomaly detection or sound detection should be independent of the choice of decision thresholds to allow for a more objective comparison [1, 3]. Because of this, metrics such as the area under the receiver-operating characteristic curve (ROC-AUC) that do not utilize any decision threshold are usually used. However, when setting up a system for practical applications detection thresholds are still needed to distinguish between normal and anomalous test samples. But without access to anomalous training samples, it is impossible to determine decision thresholds by simply testing multiple values and picking the best-performing one. Hence, estimating these thresholds is highly non-trivial and requires sophisticated techniques.

Anomalous sound detection (ASD) for machine condition monitoring is highly promoted through the annual DCASE challenge [4, 5, 6]. The baseline systems of the ASD tasks utilize the 90th percentile of a gamma distribution estimated from the histogram of the anomaly scores belonging to the normal training samples as a decision threshold. For all systems that participated in the DCASE challenge 2021 either no procedure for automatically estimating the decision thresholds is explicitly mentioned or the same (or a very similar) procedure as used by the baseline system is applied [7, 8, 9, 10, 11, 12]. The most likely reason for a lack of focus on techniques for choosing decision thresholds is that the evaluation of the ASD systems is based on the AUC

score and thus no decision thresholds are needed. This is done to have an objective comparison between the ASD performance of the systems and to prevent participants from cheating by utilizing anomalous samples of the development set for estimating thresholds. Furthermore, challenges are usually carried out only for research purposes without the goal of obtaining a fully functioning ASD system for a real-world application that would inevitably need a sophisticated technique for determining a decision threshold.

The goal of this work is to investigate multiple techniques for estimating decision thresholds in the context of anomalous sound detection for machine condition monitoring. For this purpose, first the state-of-the-art in ASD including a specific system for experimental evaluations is briefly reviewed. Second, multiple methods for estimating a decision threshold are presented. In experimental evaluations on the DCASE 2020 ASD dataset [4], these techniques are applied and compared to each other.

2 STATE-OF-THE-ART OF ANOMALOUS SOUND DETECTION

2.1 Review

First, the state-of-the-art in ASD will be reviewed. For this purpose, we will mainly follow [5]. There are two general state-of-the-art ASD paradigms for machine condition monitoring. Both rely on deep learning. The first one consists of using an autoencoder trained on normal data only. Here, it is assumed that the autoencoder can reconstruct normal data better than anomalous data due to deviations from the normal data used for training the model and thus the reconstruction error can be used as an anomaly score. Many different autoencoder architectures have been used for this purpose, e.g. class-conditioned autoencoders [13] or group masked autoencoders [14]. This approach of directly estimating the distribution of normal data is also more generally called inlier modelling (IM).

The second approach is to train a discriminative model to learn meaningful embeddings of the data. Here, it is assumed that the information needed to discriminate among predefined classes also captures the information needed to detect anomalous samples. This approach is called outlier exposure (OE) [15]. In machine condition monitoring, most models are trained to discriminate among different machine types or even finer subdivisions of the data such as different machine states or noise types [11]. To train an OE model, usually angular margin losses such as ArcFace [16] or AdaCos [17] are used [18, 19]. These losses ensure that not only inter-class similarity is minimized but simultaneously maximize intra-class similarity using an angular margin in combination with the cosine distance. Thus after training, embeddings belonging to normal samples of a specific class are concentrated around a learned mean embedding and anomalous samples are expected to have a larger angle than normal samples to this mean enabling the detection of anomalies.

Many state-of-the-art systems utilize both ASD paradigms. As noted in [5], there are two different ways to combine both approaches: a parallel and a sequential approach. The parallel approach is simply an ensemble of multiple OE and IM models [20, 21, 22] and the sequential approach consists of first applying an OE model as a feature extractor and then using an IM model for these features [23, 11]. Compared to a parallel approach, a sequential approach has the advantage that the system consists of fewer hyperparameters. However, when training a discriminative model to extract features some information needed to detect anomalous data may be lost if this information is not important for identifying the pre-defined classes.

2.2 Used system

For all experimental evaluations in this work, the system presented in [24] is used. The system is a sequential approach consisting of a neural network trained to extract discriminative audio embeddings from log-mel spectrograms using the sub-cluster AdaCos loss and a GMM for IM. The sub-cluster AdaCos loss is an extension of the AdaCos loss specifically designed for ASD. This means that the loss is also an angular margin loss with an adaptive scale parameter. The major difference to the standard AdaCos loss is that instead of learning a single class center for each class, the loss learns multiple sub-clusters for each class to learn more complex distributions. In this case, the classes are defined as specific machines recorded in noisy environments (see Section 4.1). More details about the sub-cluster AdaCos loss can be found in [24]. For all experiments, 32 sub-clusters for each class are used. When computing the log-mel spectrograms 128 mel bins, a window size of 1024 and a hop size of 512 are used.

Table 1. Modified ResNet architecture used for extracting discriminative embeddings.

layer name	structure	output size
input	-	313×128
2D convolution	7×7 , stride= 2	$157 \times 64 \times 16$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$78 \times 31 \times 16$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$39 \times 16 \times 32$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$20 \times 8 \times 64$
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2$, stride= 1	$10 \times 4 \times 128$
max pooling	10×1 , stride= 1	4×128
flatten	-	512
dense (representation)	linear	128
sub-cluster AdaCos	32 sub-clusters	41

The neural network has a modified ResNet architecture [25] as shown in Table 1 and is implemented in Tensorflow [26]. The model is trained for 400 epochs with a batch size of 64 using Adam [27]. For data augmentation, mixup [28] with a uniformly sampled mixing coefficient is used to randomly generate additional data and prevent overfitting of the model. After training, the model is used to extract embeddings, which are length-normalized by projecting them to the unit sphere. The only exception is the machine type “ToyConveyor” for which the temporal means of the log-mel spectrograms are used instead of the embeddings because these representations yield a much better ASD performance for this machine type [24]. For each pre-defined class i.e. for each specific machine of a given machine type, the distributions of the resulting embeddings are then estimated with Gaussian mixture models (GMMs) with 32 Gaussian components and a full covariance matrix, which is regularized by adding 0.001 to the diagonal as implemented in scikit-learn [29]. To obtain anomaly scores, the corresponding log-likelihood values of the GMMs are used.

3 FINDING DECISION THRESHOLDS

There are many different approaches for finding decision thresholds [30, 31]. For arbitrary anomaly scores obtained with supervised classifiers potentially being biased, these anomaly scores can be calibrated by converting them into (pseudo-)probabilities for which thresholds can be determined [32, 33]. Usually, a probability of 0.5 is used as a threshold for these calibrated scores. Since the anomaly scores of the used system already are log-likelihood values, a mapping to probabilities is not necessary. For multivariate data or scores, a threshold can be determined by using the empirical distribution function of the squared Mahalanobis distance and using a critical value such as a small quantile of the chi-squared distribution, which is the theoretical distribution function, as a threshold. [34]. An extension of this approach uses an adaptive threshold [35]. However, anomaly scores are usually univariate and thus multivariate approaches are not suitable. When continuously monitoring data streams for anomalies, these data streams themselves consist of sequential samples of which most samples are normal and only a few are anomalous. Therefore, thresholds can utilize previously encountered samples and need to adapt to changes occurring in the data stream [36, 37, 38]. This is very different from the ASD setting investigated in this work where individual recordings are either entirely normal or anomalous.

Now, several techniques for automatically estimating decision thresholds using only scores obtained with normal data will be reviewed. The general idea of all methods is to estimate a threshold that separates the extreme values of the training scores from the rest. Most of these methods are based on the assumption that the considered data, i.e. the anomaly scores, follow some specific distribution, typically a normal distribution. As this is not true for anomaly scores in general, the considered methods may work only to some extent.

3.1 Gamma distribution percentile (GP)

The strategy used by the baseline systems of the DCASE challenge [5, 6] is to fit a gamma distribution to the scores obtained with the normal training samples and use the inverse of the 90th percentile of the cumulative distribution function as the decision threshold. Test scores larger than this threshold are marked as anomalous; otherwise they are considered normal.

3.2 Histogram percentile (HP)

One can also directly use the histogram of the scores without fitting a distribution first. Note that this silently assumes a uniform distribution. We used the 90th percentile of the histogram of the scores as the decision threshold as done in [11].

3.3 Standard Deviation (SD)

One of the most commonly used approaches is to fit a normal distribution to the scores. All values exceeding the range $\mu \pm \alpha * \sigma$ are marked as anomalous, where μ and σ are the mean and standard deviation of the normal scores. Note, that technically this implies the usage of two thresholds. Since the anomaly scores in this work consist of negative log-likelihoods and thus scores belonging to normal and anomalous samples are assumed to be linearly separable, only the upper threshold is used. To have a consistent evaluation with the previous two approaches, we used $\alpha = 1.28$, which approximately corresponds to the 90th percentile.

3.4 Median Absolute Deviation (MAD)

Following the assumption that the median is more robust against outliers than the mean, decision thresholds may be obtained by $\tilde{x} \pm \alpha * \text{MAD}$, where MAD is given by $\text{MAD} = \beta * \text{median}(|x - \tilde{x}|)$ and \tilde{x} is the median value of the score values x . [31] proposes $\alpha = 3$ and $\beta = 1.4826$ following [39], [30] uses $\alpha = 2$, which is also the value we used.

3.5 Interquartile Range (IQR)

This approach is based on the division of the score values x into subsets by setting Q1 and Q3 such that $x \geq Q1$ for 75% and $x \geq Q3$ for 25% of x . Then $\text{IQR} = Q3 - Q1$. Values outside the range $Q1 - \alpha * \text{IQR}$ and $Q3 + \alpha * \text{IQR}$ are considered anomalous. Typically, $\alpha = 1.5$ is assumed [39]. We used, $\alpha = 0.5$ as this significantly improved the performance. This approach is also known as *boxplot* [30].

3.6 One-class support vector machine (OCSVM)

To estimate the support of a distribution, a one-class support vector machine [40], which learns to discriminate between regions of high and low density using a hypersphere in high-dimensional space, can also be used. We used the implementation of scikit-learn [29] with a linear kernel. For the hyperparameter ν , we used a value of 0.1 i.e. 10% of the normal training scores are treated as anomalous.

3.7 Generalized Extreme Studentized Deviate (GESD)

GESD [41] is an iterative approach based on the Grubbs's test (GRUBBS) [42]. This statistical test, named after Grubbs, assumes a normal distribution and is calculated on the so-called Grubbs statistic

$$G = \frac{|\max(x) - \mu|}{\sigma} \quad (1)$$

with mean μ and standard deviation σ . G is evaluated against a critical value of the student's t -distribution with a significance level α , set to 0.05 as default, and data size N :

$$G > \frac{N-1}{\sqrt{N}} \sqrt{\frac{t_{\alpha/(2N), N-2}^2}{N-2 + t_{\alpha/(2N), N-2}^2}}. \quad (2)$$

GRUBBS only tests for a single anomalous sample. For GESD, GRUBBS is thus repeated iteratively until no further anomalies are detected.

3.8 Clever Standard Deviation (cleverSD)

CleverSD [43] is another iterative approach. The idea is to repeatedly eliminate a single sample with the highest score from the training scores in case it is found to be anomalous by applying SD ($\alpha = 2$). This is done until no additional anomaly is found. We used the last score removed by this approach as the decision threshold.

3.9 Two-stage Thresholding (-x2)

Generalizing cleverSD, [31] suggests yet another iterative anomaly detection method called multi-stage thresholding. The main idea is to simply apply a non-iterative method multiple times to remove anomalies from the training scores. The difference to cleverSD is, that not only one anomaly but all anomalies detected are removed in each iteration. Experiments have shown that two stages are sufficient and thus we only used two iterations. When applying this technique to non-iterative approaches, we use the name of the method with the suffix -x2 to denote its two-stage version.

4 EXPERIMENTS

4.1 Dataset

For all experiments in this work, the DCASE 2020 ASD dataset [4] has been used. It consists of recordings from six different machine types, namely “ToyCar” and “ToyConveyor” from ToyAdmos [44], and “fan”, “pump”, “slider” and “valve” from MIMII [45]. Each recording contains a specific machine sound as well as factory noise and has a length of 10 seconds with a sampling rate of 16 kHz. There are six to seven different machine ids per machine type that correspond to a specific machine of that type and a total of 42 machine ids. These machine ids are used as classes when training the discriminative model described in Section 2.2.

The dataset is divided into a training set, a development set and an evaluation set. The training set consists of approximately 1000 normal sounds for each machine id. The development set consists of 100 to 200 normal sounds and 100 to 200 anomalous sounds for each of one half of the machine ids belonging to each machine type. The evaluation set consists of approximately 400 recordings containing both normal and anomalous sounds for each of the other half of machine ids.

4.2 Comparison of the decision methods

The scores obtained with the normal training data are used to estimate thresholds for ASD scores for each machine type and each machine id individually. These thresholds or the models representing them are evaluated both on the development and the evaluation set by applying them to the corresponding test sets containing a mixture of normal and anomalous samples. F1 scores are then calculated for each machine id individually and the mean is calculated for each machine type. For comparison, the performance obtained with a single optimal threshold is evaluated as an additional method denoted by *optimum*. These optimal thresholds are calculated by simply trying multiple values as decision thresholds, calculating the corresponding F1-scores and denoting the highest achieved F1-score for each machine type. To have a more robust estimation of all results, the ASD system is trained five times and the whole evaluation procedure is repeated five times for each method. Then, the mean of the five resulting F1 scores is calculated as the final performance. The final results for development and evaluation set are listed in Table 2 and Table 3, respectively. The best method for estimating decision thresholds per machine type is underlined. Additionally, average F1 scores computed over all machines types are provided. The results show that different threshold detection methods yield varying performances for distinct machine types.

To compare the different methods while reducing the influence of the difference in performance for individual machine types, we used the ratio between F1-score of a method and the best possible F1-score obtained with a single threshold, i.e. the results obtained with *optimum*, instead of the F1-scores themselves. Therefore, these values show how close an estimated threshold is to the optimal threshold and allows a better comparison between the methods regardless of the actual ASD performance of the used system for different machine types.

The results are depicted in Figure 1. The following observations can be made. First, most approaches result in a very similar ASD performance. The only exceptions are GESD, which performs worse on both the development and evaluation set, and GP/GPx2, which performs slightly worse on the development set, than

Table 2. Mean of F1 scores among all machine ids belonging to single machine types obtained with five independent trials on the development dataset. Highest F1 score among different methods for each machine type is underlined.

	fan	pump	slider	ToyCar	ToyConveyor	valve	mean
GP	0.83204	0.82253	0.91109	0.80763	0.61818	0.87869	0.81169
HP	0.75288	0.83735	0.95056	0.84823	0.67279	0.89828	0.82668
SD	0.75055	0.83927	0.95147	0.85007	0.67236	0.89985	0.82726
MAD	0.76191	0.84408	0.95138	0.85412	0.66953	<u>0.90171</u>	0.83046
IQR	0.78953	0.84310	0.93899	0.83723	0.67722	0.89694	0.83050
OCSVM	0.75288	0.83735	0.95049	0.84907	0.67265	0.89794	0.82673
GESD	0.66239	0.81641	<u>0.96752</u>	<u>0.86416</u>	0.59373	0.86991	0.79569
cleverSD	0.83559	0.83511	0.91850	0.82063	0.66289	0.88529	0.82633
GPx2	<u>0.86353</u>	0.80954	0.88840	0.77259	0.61148	0.86485	0.80173
HPx2	0.81633	0.83612	0.92617	0.81426	0.66316	0.88786	0.82398
SDx2	0.82656	0.83333	0.91861	0.80603	0.65868	0.88201	0.82087
MADx2	0.79681	<u>0.84887</u>	0.94039	0.84690	<u>0.67556</u>	0.89848	<u>0.83450</u>
IQRx2	0.83306	0.83445	0.91598	0.80581	0.65856	0.88336	0.82187
OCSVMx2	0.81573	0.83616	0.92650	0.81474	0.66307	0.88739	0.82393
optimum	0.92574	0.88895	0.98461	0.89175	0.68857	0.91896	0.88310

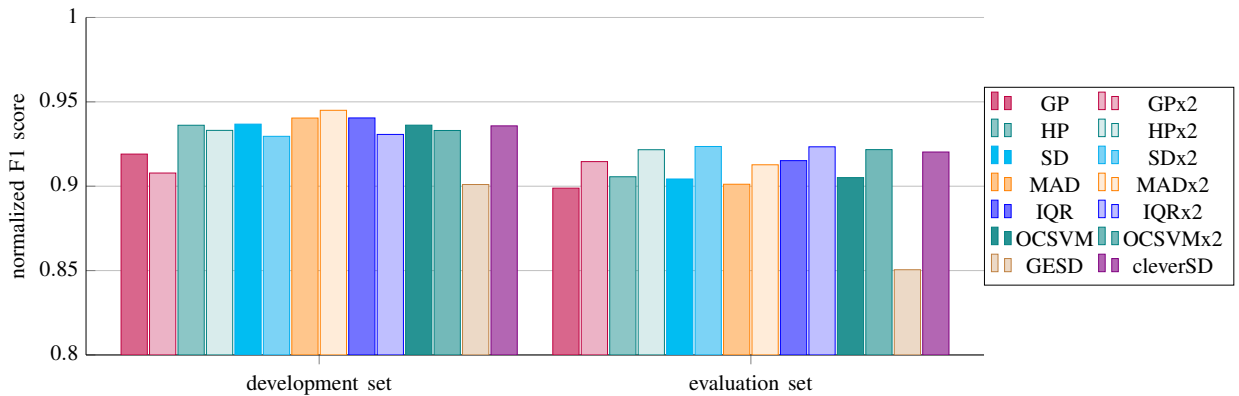


Figure 1. Comparison of decision methods based on the mean of the normalized F1 scores taken over all machine types.

all other methods. Second, the two-stage versions of the approaches, yield about the same performance on the development set and a slightly better performance on the evaluation set. Furthermore, the iterative approach cleverSD has approximately the same F1 score as the two-stage approaches. In conclusion, these experiments indicate that one should use a two-stage approach (or cleverSD) when estimating decision thresholds for ASD and the particular choice for the underlying one-stage method is not that important.

Table 3. Mean of F1 scores among all machine ids belonging to single machine types obtained with five independent trials on the evaluation dataset. Highest F1 score among different methods for each machine type is underlined.

	fan	pump	slider	ToyCar	ToyConveyor	valve	mean
GP	0.89956	0.86065	0.88863	0.60121	0.63508	0.67127	0.75940
HP	0.93845	0.89173	<u>0.92304</u>	0.58433	0.59703	0.65612	0.76512
SD	0.93675	0.89318	0.92132	0.57726	0.59974	0.65573	0.76399
MAD	0.93878	0.88953	0.91896	0.57180	0.59806	0.65103	0.76136
IQR	0.94128	<u>0.89359</u>	0.92027	0.59976	0.61941	0.66482	0.77319
OCSVM	0.93832	0.89185	0.92321	0.58277	0.59607	0.65587	0.76468
GESD	0.90994	0.88218	0.89995	0.51314	0.49565	0.61057	0.71857
cleverSD	0.94124	0.87428	0.90940	0.61826	0.64311	0.67869	0.77749
GPx2	0.90652	0.85206	0.88491	<u>0.63815</u>	<u>0.66252</u>	<u>0.69189</u>	0.77267
HPx2	0.94287	0.88472	0.91462	0.61838	0.63550	0.67583	0.77865
SDx2	0.94289	0.87921	0.91107	0.62399	0.64169	0.68273	<u>0.78026</u>
MADx2	0.94149	0.89154	0.91883	0.59180	0.61854	0.66447	0.77111
IQRx2	<u>0.94292</u>	0.87495	0.90981	0.62575	0.64314	0.68402	0.78010
OCSVMx2	0.94283	0.88521	0.91462	0.61806	0.63553	0.67594	0.77870
optimum	0.96516	0.94371	0.95913	0.75889	0.72512	0.78359	0.85593

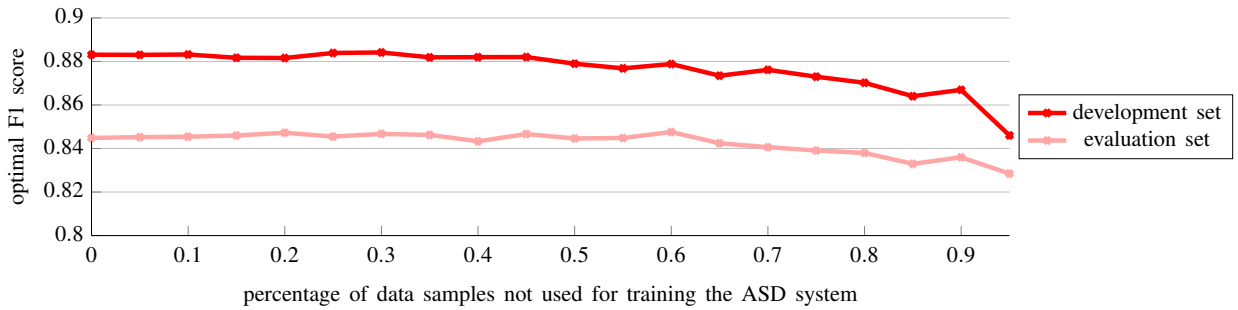


Figure 2. Mean of optimal F1 scores among machine types obtained with varying percentage of data samples not used for training the ASD system on the development set.

4.3 Dividing normal samples in disjoint sets for training ASD system and estimating decision threshold

To avoid a degraded ASD performance due to overfitting resulting from using the normal samples for estimating the decision threshold *and* training the ASD system, a commonly applied strategy is to only use a part of the normal samples for training the model and use the remaining samples for extracting more realistic scores. By using this strategy, the training scores and test scores are more similar and thus the decision threshold is expected to be more accurate. However, it is clear that using less data for training the ASD system also degrades the ASD performance since less information is incorporated into the model. In the following experiments, we investigate whether this strategy actually improves the ASD performance.

First, we evaluated the ASD performance obtained with a single optimal decision threshold for a varying number of data samples used for training the ASD system. The resulting F1 scores can be found in Figure 2. As

expected, the F1 scores decrease when using less data for training. However, the degradation in performance is much less severe than anticipated and is not noticeable before using less than 40% of normal training samples. Even when using only 5% of normal training samples, the F1 scores are only slightly worse than when using all samples. The most likely reason for this is that, ignoring the background noise, the variability of sounds emitted by machines is relatively low and thus their acoustic behavior can be captured with only a few recordings. Since this opens the possibility to train an ASD system for machine condition monitoring with much fewer computational and data resources, this observation is interesting on its own.

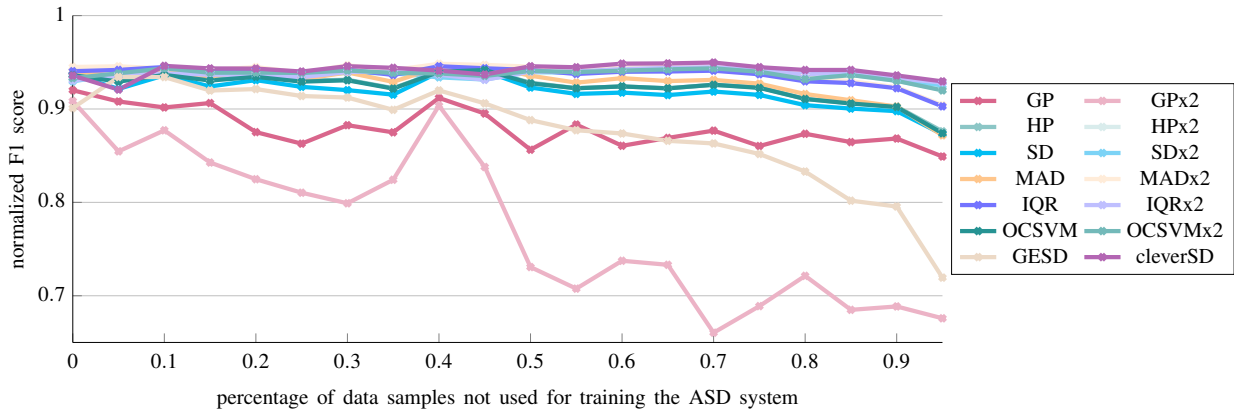


Figure 3. Mean of normalized F1 scores among machine types obtained with different methods for estimating decision thresholds and varying percentage of data samples not used for training the ASD system on the development set.

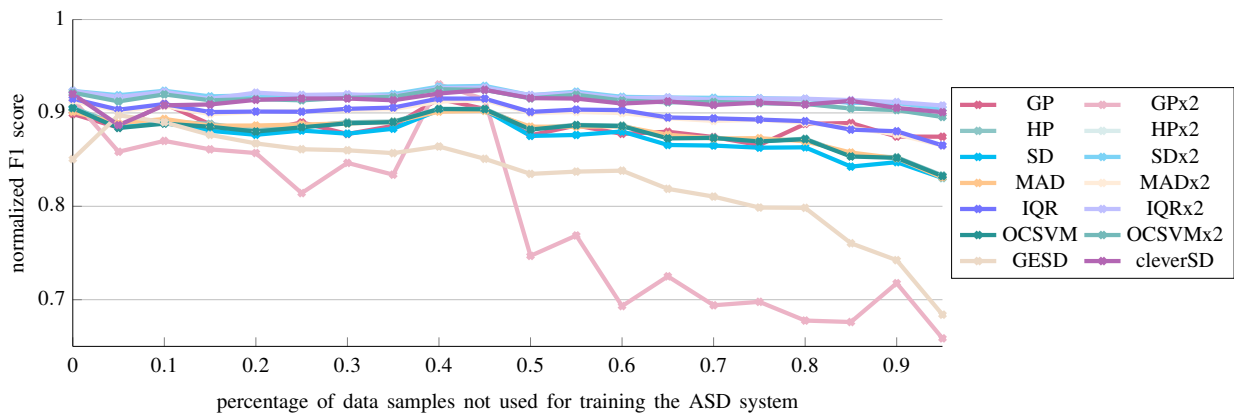


Figure 4. Mean of normalized F1 scores among machine types obtained with different methods for estimating decision thresholds and varying percentage of data samples not used for training the ASD system on the evaluation set.

Next, we evaluated the ASD performance obtained with different methods for estimating a decision threshold for a varying number of data samples used for training the ASD system. The results for the development set and evaluation set can be found in Figure 3 and Figure 4, respectively. It can be seen that using fewer samples for training the system and using these samples to estimate more realistic anomaly scores for estimating the decision threshold does not significantly improve the ASD performance. Moreover, since the absolute F1 score is actually slightly decreasing (see Figure 2) the performance is actually worse. When comparing individual methods, one can see that in general the gaps in performance get wider the less data is used for training

the ASD system. Once more, iterative approaches, namely SDx2, IQRx2, OCSVMx2 and cleverSD, perform best. Furthermore, their relative performance is relatively stable, making them a robust choice for estimating decision thresholds for ASD. Note that OCSVMx2 does not assume an underlying distribution but only linear separability of the anomaly scores. Hence, it appears to be likely that these methods, especially OCSVMx2, also work well in other settings with other ASD systems and different anomaly scores. One noticeable exception is GP, for which GPx2 the results are very noisy and much worse than every other method. Since this degraded performance is not visible to this extent when using all samples for training the ASD system, this indicates that in general the other iterative methods may be preferable.

5 CONCLUSIONS

In this work, multiple techniques for estimating decision thresholds have been reviewed and applied for detecting anomalous sounds in machine condition monitoring. In experiments conducted on the DCASE 2020 dataset, these techniques have been compared using the anomaly scores obtained with a state-of-the-art ASD system. For this particular experimental setup, the following observations have been made: First, most techniques for estimating a decision threshold perform equally well and yield approximately 90% to 95% of the F1 score obtained with an optimally tuned decision threshold. Hence, there is still a gap in performance but this gap is relatively small. Second, iterative approaches such as multi-stage thresholding [31] slightly improve the overall ASD performance and therefore are to be preferred over single-stage techniques. This is especially true when using less data for training the ASD system indicating that iterative approaches are more robust. Last but not least, holding back normal training samples (i.e. not using them for training the ASD system) for the sole purpose of obtaining more realistic anomaly scores from these samples and using the resulting scores when estimating a decision threshold does not improve the ASD performance and thus can be omitted.

Although this work is not and cannot be exhaustive in listing and comparing all methods for automatically estimating decision thresholds, it shall serve as an initial investigation on applying these techniques for practical ASD applications such as machine condition monitoring. For future work, further studies using other ASD systems for calculating the anomaly scores, other ASD datasets and additional techniques for estimating decision thresholds are to be carried out. In addition, it is planned to evaluate all mentioned methods for finding decision thresholds when dealing with domain shifts [5] and when generalizing models for multiple domains [6]. Furthermore, additional investigations on choosing decision thresholds can be led for open-set classification problems such as acoustic scene classification [46] or speaker recognition [47].

REFERENCES

- [1] Aggarwal CC. *Outlier Analysis*. Springer; 2017.
- [2] Zimek A, Filzmoser P. There and back again: Outlier detection between statistical reasoning and data mining algorithms. *WIREs Data Mining Knowl Discov*. 2018;8(6).
- [3] Ebberts J, Haeb-Umbach R, Serizel R. Threshold Independent Evaluation of Sound Event Detection Scores. In: *Proc International Conference on Acoustics, Speech and Signal Processing; 23-27 May 2022; Virtual and Singapore, China*. IEEE; 2022. p. 1021-5.
- [4] Koizumi Y, Kawaguchi Y, Imoto K, Nakamura T, Nikaido Y, Tanabe R, et al. Description and Discussion on DCASE2020 Challenge Task2: Unsupervised Anomalous Sound Detection for Machine Condition Monitoring. In: *Proc Detection and Classification of Acoustic Scenes and Events Workshop; 2-4 November 2020; Tokyo, Japan; 2020*. p. 81-5.
- [5] Kawaguchi Y, Imoto K, Koizumi Y, Harada N, Niizumi D, Dohi K, et al. Description and Discussion on DCASE 2021 Challenge Task 2: Unsupervised Anomalous Detection for Machine Condition Monitoring Under Domain Shifted Conditions. In: *Proc Detection and Classification of Acoustic Scenes and Events Workshop; 15-19 November 2021; Online; 2021*. p. 186-90.

- [6] Dohi K, Imoto K, Harada N, Niizumi D, Koizumi Y, Nishida T, et al. Description and Discussion on DCASE 2022 Challenge Task 2: Unsupervised Anomalous Sound Detection for Machine Condition Monitoring Applying Domain Generalization Techniques. In arXiv e-prints: 220605876. 2022.
- [7] Bai J, Wang M, Chen J. Dual-Path Transformer For Machine Condition Monitoring. In: Proc Asia-Pacific Signal and Information Processing Association Annual Summit and Conference; 14-17 December 2021; Tokyo, Japan. IEEE; 2021. p. 1144-8.
- [8] Li R, Gu X, Lu F, Song H, Pan J. Unsupervised Adversarial domain adaptive abnormal sound detection for machine condition monitoring under Domain Shift Conditions. DCASE2021 Challenge; Tech Rep; 2021.
- [9] Narita H, Tamamori A. Unsupervised Anomalous Sound Detection Using Intermediate Representation of Trained Models and Metric Learning Based Variational Autoencoder. DCASE2021 Challenge; Tech Rep; 2021.
- [10] Pham L, Jalali A, Dinica O, Schindler A. DCASE Challenge 2021: Unsupervised Anomalous Sound Detection of Machinery with LeNet Architecture. DCASE2021 Challenge; Tech Rep; 2021.
- [11] Wilkinghoff K. Combining Multiple Distributions based on Sub-Cluster AdaCos for Anomalous Sound Detection under Domain Shifted Conditions. In: Proc Detection and Classification of Acoustic Scenes and Events Workshop; 15-19 November 2021; Online; 2021. p. 55-9.
- [12] Zhang C, Yao Y, Qiu R, Li S, Shao X. Unsupervised Anomalous Sound Detection Using Denoising-Detection System Under Domain Shifted Conditions. DCASE2021 Challenge; Tech Rep; 2021.
- [13] Kapka S. ID-Conditioned Auto-Encoder for Unsupervised Anomaly Detection. In: Proc 5th Workshop on Detection and Classification of Acoustic Scenes and Events; 2-4 November 2020; Tokyo, Japan (full virtual); 2020. p. 71-5.
- [14] Giri R, Cheng F, Helwani K, Tenneti SV, Isik U, Krishnaswamy A. Group Masked Autoencoder Based Density Estimator for Audio Anomaly Detection. In: Proc 5th Workshop on Detection and Classification of Acoustic Scenes and Events; 2-4 November 2020; Tokyo, Japan (full virtual); 2020. p. 51-5.
- [15] Hendrycks D, Mazeika M, Dietterich TG. Deep Anomaly Detection with Outlier Exposure. In: Proc 7th International Conference on Learning Representations; 6-9 May 2019; New Orleans, LA, USA. OpenReview.net; 2019. p. 1-18.
- [16] Deng J, Guo J, Xue N, Zafeiriou S. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In: Conference on Computer Vision and Pattern Recognition; 16-20 June 2019; Long Beach, CA, USA. IEEE; 2019. p. 4690-9.
- [17] Zhang X, Zhao R, Qiao Y, Wang X, Li H. AdaCos: Adaptively Scaling Cosine Logits for Effectively Learning Deep Face Representations. In: Proc Conference on Computer Vision and Pattern Recognition; 16-20 June 2019; Long Beach, CA, USA. IEEE; 2019. p. 10823-32.
- [18] Zhou Q. ArcFace Based Sound Mobilenets for DCASE 2020 task 2. DCASE2020 Challenge; Tech Rep; 2020.
- [19] Lopez JA, Lu H, Lopez-Meyer P, Nachman L, Stemmer G, Huang J. A Speaker Recognition Approach to Anomaly Detection. In: Proc 5th Workshop on Detection and Classification of Acoustic Scenes and Events; 2-4 November 2020; Tokyo, Japan (full virtual); 2020. p. 96-9.
- [20] Lopez JA, Stemmer G, Lopez-Meyer P, Singh P, del Hoyo Ontiveros JA, Cordourier HA. Ensemble Of Complementary Anomaly Detectors Under Domain Shifted Conditions. In: Proc Detection and Classification of Acoustic Scenes and Events Workshop; 15-19 November 2021; Online; 2021. p. 11-5.

- [21] Kuroyanagi I, Hayashi T, Adachi Y, Yoshimura T, Takeda K, Toda T. An Ensemble Approach to Anomalous Sound Detection Based on Conformer-Based Autoencoder and Binary Classifier Incorporated with Metric Learning. In: Proc Detection and Classification of Acoustic Scenes and Events Workshop; 15-19 November 2021; Online; 2021. p. 110-4.
- [22] Sakamoto Y, Miyamoto N. Combine Mahalanobis Distance, Interpolation Auto Encoder and Classification Approach for Anomaly Detection. DCASE2021 Challenge; Tech Rep; 2021.
- [23] Morita K, Yano T, Tran K. Anomalous Sound Detection Using CNN-Based Features By Self Supervised Learning. DCASE2021 Challenge; Tech Rep; 2021.
- [24] Wilkinghoff K. Sub-Cluster AdaCos: Learning Representations for Anomalous Sound Detection. In: Proc International Joint Conference on Neural Networks; 18-22 July 2021; Shenzhen, China. IEEE; 2021. p. 1-8.
- [25] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. In: Proc Conference on Computer Vision and Pattern Recognition; 27-30 June 27-30 2016; Las Vegas, NV, USA. IEEE; 2016. p. 770-8.
- [26] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. Tensorflow: A system for large-scale machine learning. In: Proc 12th USENIX Symposium on Operating Systems Design and Implementation; 2-4 November 2016; Savannah, GA, USA; 2016. p. 265-83.
- [27] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. In: Proc 3rd International Conference on Learning Representations; 7-9 May 2015; San Diego, CA, USA; 2015. p. 1-15.
- [28] Zhang H, Cisse M, Dauphin YN, Lopez-Paz D. Mixup: Beyond empirical risk minimization. In: Proc International Conference on Learning Representations; 30 April - 3 May 2018; Vancouver, BC, Canada; 2018. p. 1-13.
- [29] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011;12:2825-30.
- [30] Reimann C, Filzmoser P, Garrett RG. Background and threshold: critical comparison of methods of determination. *Science of the total environment*. 2005;346(1-3):1-16.
- [31] Yang J, Rahardja S, Fränti P. Outlier detection: How to threshold outlier scores? In: Proc International Conference on Artificial Intelligence, Information Processing and Cloud Computing; 19-21 December 2019; Sanya, China. ACM; 2019. p. 37:1-37:6.
- [32] Gao J, Tan P. Converting Output Scores from Outlier Detection Algorithms into Probability Estimates. In: Proc 6th International Conference on Data Mining; 18-22 December 2006; Hong Kong, China. IEEE; 2006. p. 212-21.
- [33] Gebel M. Multivariate calibration of classifier scores into the probability space [Ph.D. thesis]. University of Dortmund; 2009.
- [34] Rousseeuw PJ, Van Zomeren BC. Unmasking multivariate outliers and leverage points. *Journal of the American Statistical association*. 1990;85(411):633-9.
- [35] Filzmoser P. A multivariate outlier detection method. In: Proc 7th International Conference on Computer Data Analysis and Modeling; Minsk, Belarus; 2004. p. 18-22.
- [36] Clark J, Liu Z, Japkowicz N. Adaptive Threshold for Outlier Detection on Data Streams. In: Proc 5th International Conference on Data Science and Advanced Analytics; 1-3 October 2018; Turin, Italy. IEEE; 2018. p. 41-9.

- [37] Gökcesu K, Neyshabouri MM, Gökcesu H, Kozat SS. Sequential Outlier Detection Based on Incremental Decision Trees. *IEEE Trans Signal Process.* 2019;67(4):993-1005.
- [38] Zhang M, Li X, Wang L. An Adaptive Outlier Detection and Processing Approach Towards Time Series Sensor Data. *IEEE Access.* 2019;7:175192-212.
- [39] Rousseeuw PJ, Croux C. Alternatives to the median absolute deviation. *Journal of the American Statistical association.* 1993;88(424):1273-83.
- [40] Schölkopf B, Platt JC, Shawe-Taylor J, Smola AJ, Williamson RC. Estimating the Support of a High-Dimensional Distribution. *Neural Comput.* 2001;13(7):1443-71.
- [41] Rosner B. Percentage Points for a Generalized ESD Many-Outlier Procedure. *Technometrics.* 1983;25(2):165-72.
- [42] Grubbs FE. Sample Criteria for Testing Outlying Observations. *The Annals of Mathematical Statistics.* 1950;21(1):27-58.
- [43] Buzzi-Ferraris G, Manenti F. Outlier detection in large data sets. *Computers & chemical engineering.* 2011;35(2):388-90.
- [44] Koizumi Y, Saito S, Uematsu H, Harada N, Imoto K. ToyADMOS: A Dataset of Miniature-Machine Operating Sounds for Anomalous Sound Detection. In: *Proc Workshop on Applications of Signal Processing to Audio and Acoustics*; October 20-23, 2019; New Paltz, NY, USA. IEEE; 2019. p. 313-7.
- [45] Purohit H, Tanabe R, Ichige T, Endo T, Nikaido Y, Suefusa K, et al. MIMII Dataset: Sound Dataset for Malfunctioning Industrial Machine Investigation and Inspection. In: *Proc Detection and Classification of Acoustic Scenes and Events Workshop*; 25-26 October 2019; New York, NY, USA; 2019. p. 209-13.
- [46] Mesaros A, Heittola T, Virtanen T. Acoustic Scene Classification in DCASE 2019 Challenge: Closed and Open Set Classification and Data Mismatch Setups. In: *Proc Workshop on Detection and Classification of Acoustic Scenes and Events*; 25-26 October 2019; New York, USA; 2019. p. 164-8.
- [47] Shon S, Dehak N, Reynolds DA, Glass JR. MCE 2018: The 1st Multi-Target Speaker Detection and Identification Challenge Evaluation. In: *Proc 20th Annual Conference of the International Speech Communication Association*; 15-19 September 2019; Graz, Austria. ISCA; 2019. p. 356-60.

A.1.4 *Key publication 4*

Kevin Wilkinghoff. “Design Choices for Learning Embeddings from Auxiliary Tasks for Domain Generalization in Anomalous Sound Detection.” In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2023. DOI: [10.1109/ICASSP49357.2023.10097176](https://doi.org/10.1109/ICASSP49357.2023.10097176).

© 2023 IEEE.

DESIGN CHOICES FOR LEARNING EMBEDDINGS FROM AUXILIARY TASKS FOR DOMAIN GENERALIZATION IN ANOMALOUS SOUND DETECTION

Kevin Wilkinghoff 

Fraunhofer FKIE, Fraunhoferstraße 20, 53343 Wachtberg, Germany
kevin.wilkinghoff@fkie.fraunhofer.de

ABSTRACT

Emitted machine sounds can change drastically due to a change in settings of machines or varying noise conditions resulting in false alarms when monitoring machine conditions with a trained anomalous sound detection (ASD) system. In this work, a conceptually simple state-of-the-art ASD system based on embeddings learned through auxiliary tasks generalizing to multiple data domains is presented. In experiments conducted on the DCASE 2022 ASD dataset, particular design choices such as preventing trivial projections, combining multiple input representations and choosing a suitable backend are shown to significantly improve the ASD performance.

Index Terms— anomalous sound detection, representation learning, domain generalization, machine listening

1. INTRODUCTION

Semi-supervised anomalous sound detection (ASD) for machine condition monitoring is the task of training an ASD system to distinguish normal from anomalous machine sounds using only normal training samples [1, 2]. To avoid the need to repeatedly collect data and retrain the ASD system in case acoustic conditions or machine attributes change, domain generalization (DG) [3] techniques can be applied. For DG, there are two data domains with somehow different acoustics: a source domain with many training samples and a target domain with only a few training samples. The goal is to obtain an ASD system that correctly detects anomalies regardless of whether sounds belong to the source or target domain.

The most popular state-of-the-art approach for ASD is to train a neural network to extract discriminative embeddings through auxiliary tasks, also called outlier exposed ASD [4], using angular margin losses such as ArcFace [5], AdaCos [6] or sub-cluster AdaCos [7]. Examples of auxiliary tasks are to discriminate between different machine types or among other acoustic characteristics such as different machine settings or noise conditions. The assumption is that the information needed for correctly classifying the sounds is also sufficient to detect anomalous sounds. In contrast to directly modeling the distribution of the data (inlier modeling), e.g. by using autoencoders, the model learns to ignore other sound

events and thus to handle noise more effectively whereas a model trained on a single class cannot tell the difference between important and irrelevant components of the signal.

The main contributions of this work are the following: First and foremost, a state-of-the-art ASD system with strong domain generalization capabilities in machine condition monitoring is presented¹. The system is conceptually simple since its architecture and all hyperparameter settings are the same for each machine type and no external data resources are used for training the system. Furthermore, in ablation studies several design choices are shown to have a significant impact on the performance in the source and target domain. These design choices are 1) preventing trivial projections to hyperspheres, 2) combining multiple input feature representations by jointly training sub-networks and 3) choosing a suitable backend for generalizing to multiple data domains.

2. SYSTEM DESCRIPTION

An overview of the ASD system is shown in Fig. 1. The system is an improved version of the one described in [8].

2.1. Frontend

The system uses two different feature representations derived from raw waveforms of 10 seconds length and a sampling rate of 16kHz as input. First, magnitude spectrograms with a Hanning-windowed DFT length of 1024 and a hop size of 512. Second, magnitude spectra of entire signals are used to have the highest possible frequency resolution for better capturing stationary sounds. To reduce acoustic differences between source and target domains, sample-wise temporal mean normalization similar to cepstral mean normalization [9] is applied to the magnitude spectrograms.

2.2. Neural network architecture

The neural network for extracting the embeddings consists of two different sub-networks for each input representation. To capture as much information as possible, the entire network

¹An open-source implementation of the proposed system is available at: <https://github.com/wilkinghoff/icassp2023>

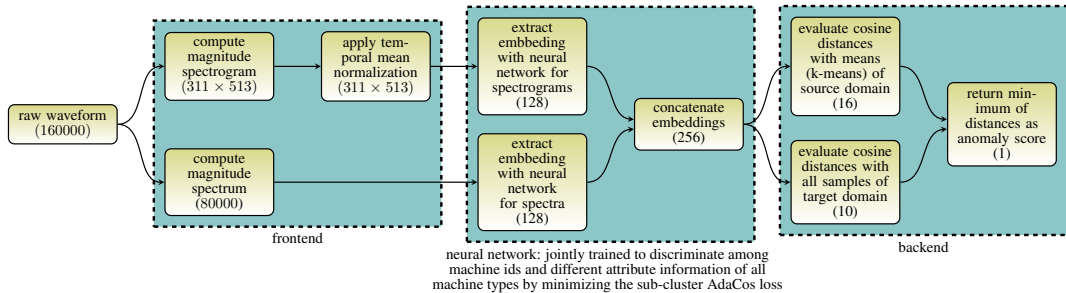


Fig. 1. Structure of the proposed anomalous sound detection system. Representation size in each step is given in brackets.

Table 1. Modified ResNet architecture for spectrograms.

layer name	structure	output size
input	BN (temporal axis)	311 × 513
2D convolution	7 × 7, stride= 2	156 × 257 × 16
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2, \text{ stride}= 1$	77 × 128 × 16
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2, \text{ stride}= 1$	39 × 64 × 32
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2, \text{ stride}= 1$	20 × 32 × 64
residual block	$\begin{pmatrix} 3 \times 3 \\ 3 \times 3 \end{pmatrix} \times 2, \text{ stride}= 1$	10 × 16 × 128
max pooling	10 × 1, stride= 1	1 × 16 × 128
flatten	BN	2056
dense (embedding)	linear	128

Table 2. Network architecture for spectra.

layer name	structure	output size
input	-	80000
1D convolution	256, stride= 64	1250 × 128
1D convolution	64, stride= 16	40 × 128
1D convolution	16, stride= 4	10 × 128
flatten	-	1280
dense	BN, ReLU	128
dense	BN, ReLU	128
dense	BN, ReLU	128
dense	BN, ReLU	128
dense (embedding)	linear	128

is trained to discriminate between all machine types, sections and different attribute information about the machines resulting in a total of 342 classes by minimizing the sub-cluster AdaCos loss [7] with 16 sub-clusters for each class. In contrast to [10], where multiple networks and loss functions have been used for different classification tasks, this training strategy is much simpler. The sub-network used for the spectrograms is based on a modified ResNet architecture [11] as also used in [7,8,10] and is described in Tab. 1. For the spectra, the sub-network consists of three one-dimensional convolutions and five dense layers as shown in Tab. 2. Note that improving the results for the auxiliary task does not necessarily imply a better ASD performance and thus adjusting the number of layers and their parameter size is not critical for improving the ASD capabilities. However, both sub-network architectures are carefully designed to avoid learning trivial mappings to hyperspheres for specific classes as done in networks for deep one-class classification [12]. This means that 1) no bounded non-linearities, 2) no bias terms and 3) no trainable hypersphere centers are used. Instead, for 1) we only use rectified linear units as non-linearities and for 3) we randomly initialize the cluster centers of the sub-cluster AdaCos loss without adapting them during training. A random initialization of the cluster centers is not a problem, since embeddings and cluster centers live in a relatively high-dimensional space and thus are very likely to be pairwise orthogonal.

The output of both sub-networks, which can both be interpreted as embeddings by themselves, are concatenated to obtain a single embedding for each file. This concatenation

ensures that *both* networks capture all information needed to discriminate between the classes present in their respective feature representations. Therefore, the embeddings are more sensitive to anomalous sounds than when giving the network the freedom to utilize only a single feature representation (e.g. by taking the sum) because specific anomalies may be apparent in only one of the two input representations.

The entire network is implemented using Tensorflow [13] and is trained for 10 epochs with a batch size of 64 using Adam [14]. For data augmentation, only mixup [15] with a uniformly distributed mixing coefficient is used.

2.3. Backend

To obtain an anomaly score for a test sample, for each domain a different strategy is applied. For the source domain, k-means with 16 clusters is applied to the embeddings obtained with all source samples and the cosine distance between the test embedding and all these means is calculated. For the target domain, the cosine distance between the test embedding and the embeddings of all target samples is calculated. Finally, the minimum over all distances is returned as an anomaly score with higher scores indicating anomalies.

3. EXPERIMENTAL SETUP

For all experiments, the DCASE 2022 ASD dataset [2] was used. This dataset consists of sounds from the machine types “bearing”, “fan”, “gearbox”, “slider”, “valve” from MIMII DG [16], and “ToyCar”, “ToyTrain” from ToyADMOS2 [17]. For each machine type, there are 6 different subsets of the

Table 3. Comparison between using or not using trainable cluster centers and bias terms.

dataset	domain	trainable cluster centers		non-trainable cluster centers	
		AUC (%)	pAUC (%)	AUC (%)	pAUC (%)
using bias terms					
dev	source	81.65 ± 0.85	70.25 ± 1.31	82.75 ± 0.95	75.22 ± 0.78
dev	target	77.18 ± 0.88	62.05 ± 1.10	76.84 ± 1.39	61.66 ± 1.09
dev	mixed	77.73 ± 0.90	64.09 ± 1.40	79.79 ± 0.50	64.89 ± 0.41
eval	source	74.21 ± 1.14	62.84 ± 1.38	76.45 ± 0.71	65.12 ± 0.66
eval	target	71.13 ± 1.24	59.54 ± 0.86	69.19 ± 0.56	59.08 ± 1.15
eval	mixed	71.91 ± 0.98	58.84 ± 0.88	73.04 ± 0.58	59.18 ± 0.90
not using bias terms					
dev	source	82.88 ± 0.68	71.44 ± 0.96	84.19 ± 0.75	76.45 ± 0.90
dev	target	77.68 ± 1.11	62.82 ± 1.17	78.51 ± 0.90	62.54 ± 0.87
dev	mixed	78.15 ± 0.77	64.86 ± 0.52	81.36 ± 0.66	66.55 ± 0.86
eval	source	74.34 ± 0.96	63.49 ± 0.38	76.81 ± 0.79	65.84 ± 0.22
eval	target	71.30 ± 0.33	59.94 ± 0.81	69.80 ± 0.53	59.67 ± 1.14
eval	mixed	72.15 ± 0.50	58.90 ± 0.42	73.43 ± 0.54	59.78 ± 0.83

dataset called *sections*, of which three belong to a development set and the other three belong to an evaluation set, corresponding to different types of domain shifts. For each section, there are 990 normal training samples belonging to the source domain, 10 normal training samples belonging to a target domain and 200 test samples each belonging to one of the domains. Furthermore, some attribute information defining states of the machines or different types of noise are given for training samples. All recordings include real factory noise, have a length of 10 seconds and a sampling rate of 16 kHz. For training the network, all machine types, sections and different attribute information about the machines belonging to the development and evaluation set (in each case both domains) are used as classes for the auxiliary task.

In each experiment, each system was trained five times and the arithmetic mean and standard deviation of the harmonic means of all AUCs and pAUCs obtained for all 42 machine ids are shown. Highest AUCs and pAUCs for each combination of dataset and domain are highlighted in bold letters.

4. EXPERIMENTAL RESULTS

4.1. Preventing trivial projections to hyperspheres

In Tab. 3, the performance obtained with trainable and non-trainable class centers, and when using or not using bias terms are compared. Although non-trainable class centers decrease the performance on the target domain, not using bias terms or trainable class centers improves the overall performance.

4.2. Input feature representations

Next, different input feature representations and their combinations are compared. The results are shown in Tab. 4 and show that magnitude spectrograms perform better than magnitude spectra and log-mel magnitude spectrograms with 128 mel bins. When combining a spectrogram with the full spectrum, the performance is even better since the model also has access to the whole frequency resolution instead of only a time-frequency representation. Moreover, the best way to

Table 4. Comparison between different input feature representations and ways of combining them.

dataset	domain	individual input feature representations					
		magnitude spectrum		log-mel magnitude spectrogram		magnitude spectrogram	
		AUC (%)	pAUC (%)	AUC (%)	pAUC (%)	AUC (%)	pAUC (%)
dev	source	80.75 ± 0.92	71.09 ± 1.14	70.91 ± 2.10	63.21 ± 1.10	79.18 ± 1.07	70.38 ± 0.43
dev	target	73.95 ± 1.25	61.31 ± 1.40	65.78 ± 1.48	57.05 ± 0.76	76.59 ± 1.59	60.59 ± 1.50
dev	mixed	76.81 ± 0.94	63.09 ± 1.23	68.93 ± 1.52	57.79 ± 0.55	77.60 ± 1.02	62.31 ± 1.10
eval	source	68.42 ± 1.02	59.06 ± 0.89	67.59 ± 1.01	59.96 ± 0.65	74.65 ± 0.83	64.04 ± 1.23
eval	target	63.46 ± 1.39	56.90 ± 0.95	62.09 ± 0.64	56.60 ± 0.79	69.67 ± 1.14	58.95 ± 0.74
eval	mixed	66.00 ± 1.11	57.11 ± 0.58	65.04 ± 0.68	57.00 ± 0.50	72.10 ± 0.81	58.87 ± 0.33
combining magnitude spectrum and magnitude spectrograms							
		concatenate embeddings		add embeddings		concatenate embeddings	
		after training		while training		while training	
dataset	domain	AUC (%)	pAUC (%)	AUC (%)	pAUC (%)	AUC (%)	pAUC (%)
dev	source	84.68 ± 0.86	74.51 ± 0.26	83.12 ± 1.18	73.25 ± 1.11	84.19 ± 0.75	76.45 ± 0.90
dev	target	78.78 ± 0.78	63.00 ± 0.36	77.96 ± 1.37	62.02 ± 1.11	78.51 ± 0.90	62.54 ± 0.87
dev	mixed	81.17 ± 0.66	65.23 ± 0.39	80.21 ± 0.73	64.40 ± 1.20	81.36 ± 0.66	66.55 ± 0.86
eval	source	75.27 ± 0.96	63.93 ± 0.63	75.54 ± 0.83	64.85 ± 0.73	76.81 ± 0.79	65.84 ± 0.22
eval	target	69.31 ± 0.90	59.45 ± 0.67	69.71 ± 0.39	59.08 ± 1.15	69.80 ± 0.53	59.67 ± 1.14
eval	mixed	72.18 ± 0.67	59.45 ± 0.46	72.48 ± 0.53	59.22 ± 0.66	73.43 ± 0.54	59.78 ± 0.83

Table 5. Effect of temporal normalization.

dataset	domain	without temporal normalization		with temporal normalization	
		AUC (%)	pAUC (%)	AUC (%)	pAUC (%)
magnitude spectrogram					
dev	source	79.93 ± 0.71	70.98 ± 1.38	79.18 ± 1.07	70.38 ± 0.43
dev	target	76.18 ± 0.85	60.35 ± 1.23	76.59 ± 1.59	60.59 ± 1.50
dev	mixed	77.69 ± 0.47	62.52 ± 1.09	77.60 ± 1.02	62.31 ± 1.10
eval	source	75.53 ± 1.19	64.31 ± 1.08	74.65 ± 0.83	64.04 ± 1.23
eval	target	69.52 ± 0.73	59.67 ± 0.71	69.67 ± 1.14	58.95 ± 0.74
eval	mixed	72.19 ± 0.77	59.39 ± 0.91	72.10 ± 0.81	58.87 ± 0.33
magnitude spectrum + magnitude spectrogram					
dev	source	83.18 ± 1.68	74.66 ± 0.71	84.19 ± 0.75	76.45 ± 0.90
dev	target	76.95 ± 0.97	62.26 ± 0.62	78.51 ± 0.90	62.54 ± 0.87
dev	mixed	80.04 ± 0.76	64.84 ± 0.51	81.36 ± 0.66	66.55 ± 0.86
eval	source	76.41 ± 0.48	65.39 ± 0.68	76.81 ± 0.79	65.84 ± 0.22
eval	target	68.89 ± 0.96	59.46 ± 0.68	69.80 ± 0.53	59.67 ± 1.14
eval	mixed	72.85 ± 0.61	59.91 ± 0.75	73.43 ± 0.54	59.78 ± 0.83

combine these representations is by concatenating the corresponding embeddings while training.

In Tab. 5, the effect of using temporal normalization for the magnitude spectrograms is investigated. It can be seen that temporal normalization improves the performance on the target domain as intended while slightly but not significantly degrading the performance on the source domain when using only magnitude spectrograms. But when combining spectra and spectrograms, temporal normalization also improves the performance on the source domain. The reason is that both input representations complement each other more effectively because stationary frequency information are removed from the spectrograms but are clearly contained in the spectra.

4.3. Backends

In Tab. 6, different backends, namely cosine distance and a GMM with 16 Gaussian components and a full covariance matrix regularized by adding 10^{-3} to the diagonal (for more details, see [8]), are compared. The following observations can be made: As expected, specialized models for individual domains perform better on the domain they are trained on and much worse on the other domain. For the source domain, the GMM model has a slightly higher performance than the cosine similarity, which is consistent with the findings in [7]. For the target domain, both backends are equivalent.

Table 6. Comparison between different backends.

dataset	domain	Gaussian mixture model (GMM)		cosine distance	
		AUC (%)	pAUC (%)	AUC (%)	pAUC (%)
using scores from source domain model only					
dev	source	82.97 ± 0.97	77.36 ± 0.38	83.10 ± 1.02	76.87 ± 0.26
dev	target	66.52 ± 0.42	59.63 ± 0.70	71.66 ± 1.25	61.45 ± 0.83
dev	mixed	71.48 ± 0.26	58.86 ± 0.38	76.72 ± 0.78	63.37 ± 0.71
eval	source	77.46 ± 1.16	66.73 ± 0.56	76.68 ± 0.85	66.25 ± 0.51
eval	target	44.23 ± 3.67	54.87 ± 0.46	57.90 ± 1.17	55.62 ± 1.24
eval	mixed	63.83 ± 0.62	55.74 ± 0.33	67.24 ± 0.70	56.83 ± 0.92
using scores from target domain model only					
dev	source	62.41 ± 2.80	60.54 ± 1.44	62.42 ± 2.80	60.55 ± 1.44
dev	target	79.93 ± 0.92	62.19 ± 1.05	79.92 ± 0.92	62.18 ± 1.06
dev	mixed	70.84 ± 1.13	58.36 ± 1.38	70.84 ± 1.13	58.36 ± 1.38
eval	source	52.82 ± 3.40	56.27 ± 1.17	52.80 ± 3.41	52.26 ± 1.17
eval	target	71.15 ± 0.50	60.72 ± 0.95	71.15 ± 0.50	60.72 ± 0.95
eval	mixed	62.55 ± 0.79	54.66 ± 0.92	62.55 ± 0.79	54.65 ± 0.92
using sum of scores from both domain models					
dev	source	75.94 ± 2.70	70.34 ± 2.69	79.44 ± 1.95	73.29 ± 2.60
dev	target	78.64 ± 1.12	63.28 ± 0.94	78.84 ± 1.23	62.67 ± 0.98
dev	mixed	77.10 ± 1.59	65.12 ± 1.63	77.83 ± 1.56	66.11 ± 1.61
eval	source	64.57 ± 1.35	60.96 ± 1.21	68.96 ± 1.11	63.09 ± 0.87
eval	target	66.69 ± 0.54	58.73 ± 0.97	67.84 ± 0.48	58.80 ± 1.33
eval	mixed	65.70 ± 0.91	57.26 ± 0.77	67.98 ± 0.64	58.12 ± 0.71
using scores from joint model for both domains					
dev	source	84.57 ± 0.70	77.57 ± 0.41	84.19 ± 0.75	76.45 ± 0.90
dev	target	77.26 ± 1.02	63.12 ± 1.24	78.51 ± 0.90	62.54 ± 0.87
dev	mixed	80.06 ± 0.35	64.67 ± 0.70	81.36 ± 0.66	66.55 ± 0.86
eval	source	78.15 ± 0.95	67.55 ± 0.63	76.81 ± 0.79	65.84 ± 0.22
eval	target	65.17 ± 0.48	58.59 ± 0.79	69.80 ± 0.53	59.67 ± 1.14
eval	mixed	71.35 ± 0.53	59.32 ± 0.83	73.43 ± 0.54	59.78 ± 0.83

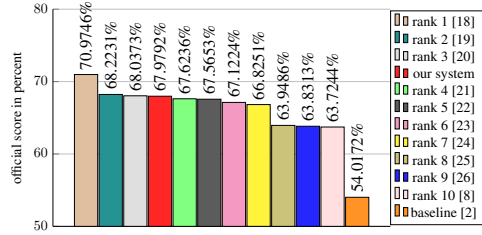
Table 7. Effect of the presented design choices.

dataset	domain	standard design choices		presented design choices	
		AUC (%)	pAUC (%)	AUC (%)	pAUC (%)
dev	source	71.65 ± 1.07	62.71 ± 0.74	84.19 ± 0.75	76.45 ± 0.90
dev	target	63.63 ± 1.11	55.28 ± 0.84	78.51 ± 0.90	62.54 ± 0.87
dev	mixed	67.72 ± 0.79	55.94 ± 0.54	81.36 ± 0.66	66.55 ± 0.86
eval	source	69.11 ± 1.05	58.46 ± 0.42	76.81 ± 0.79	65.84 ± 0.22
eval	target	61.33 ± 0.70	54.82 ± 0.78	69.80 ± 0.53	59.67 ± 1.14
eval	mixed	64.59 ± 0.74	55.22 ± 0.66	73.43 ± 0.54	59.78 ± 0.83

For domain generalization, it is necessary to use a single decision threshold for both domains and thus a single ASD score is required. It can be seen that training a joint model has a significantly higher performance than adding the ASD scores of individually trained models. However, for the target domain the joint model is worse than a specialized model. For the source domain, the joint GMM model has a better performance than using cosine distance again and interestingly even outperforms the specialized model. But when jointly evaluating both domains, using the cosine distance as a backend has a higher performance than using a GMM. The most probably reason is that in contrast to a GMM the cosine distance requires no training and thus the resulting ASD scores are scaled more consistently among both domains.

4.4. Putting it all together

To show the strong impact of the design choices on the ASD performance, we compared a system with standard design choices to the presented ones. The standard design choices are using log-mel magnitude spectrograms, trainable cluster centers, bias terms, and GMMs as backend and are for example used in the ASD system [10] ranked third in the DCASE challenge 2021. The results are shown in Tab. 7. It can be

**Fig. 2.** Comparison between presented, baseline and 10 top-performing systems of the DCASE challenge 2022.

seen that the presented design choices led to a significant performance improvement for all domains and dataset splits.

4.5. Comparison to other systems

Last, we compared the performance of our system (see Fig. 1) to the baseline and 10 top-performing systems of the DCASE challenge 2022. For these experiments, we trained the presented system five times and created an ensemble by adding the corresponding anomaly scores. The results can be found in Fig. 2. Our system would have obtained rank 4, performing close to the rank 3 system [20], and thus can be considered state-of-the-art. Note that the rank 1 system [18] used manually customized band-pass filters for each machine type without stating the details in the report and monitored the AUC score on the development set while training to find the best performing model parameters and thus also indirectly used anomalous samples for training. Both do not allow a fair comparison between performances and are a possible explanation for the large performance gap to all other systems.

5. CONCLUSIONS

In this work a conceptually simple ASD system based on learning embeddings through auxiliary tasks for domain generalization was presented. In various experiments conducted on the DCASE 2022 ASD dataset, it was shown that several design choices, namely preventing trivial projections, utilizing multiple input feature representations and choosing a suitable backend, significantly improve the performance and that the presented system achieves state-of-the-art performance. For future work, it is planned to carry out additional experiments to verify whether the presented design choices also improve the performances when using other ASD systems.

6. REFERENCES

- [1] Yohei Kawaguchi et al., “Description and discussion on DCASE 2021 challenge task 2: Unsupervised anomalous detection for machine condition monitoring under domain shifted conditions,” in *DCASE*, 2021, pp. 186–190.

- [2] Kota Dohi et al., “Description and discussion on DCASE 2022 challenge task 2: Unsupervised anomalous sound detection for machine condition monitoring applying domain generalization techniques,” in *DCASE*. 2022, pp. 26–30, Tampere University.
- [3] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, and Tao Qin, “Generalizing to unseen domains: A survey on domain generalization,” in *IJCAI*. 2021, pp. 4627–4635, ijcai.org.
- [4] Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich, “Deep anomaly detection with outlier exposure,” in *ICLR*. 2019, OpenReview.net.
- [5] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, “ArcFace: Additive angular margin loss for deep face recognition,” in *CVPR*. 2019, pp. 4690–4699, IEEE.
- [6] Xiao Zhang, Rui Zhao, Yu Qiao, Xiaogang Wang, and Hongsheng Li, “AdaCos: Adaptively scaling cosine logits for effectively learning deep face representations,” in *CVPR*. 2019, pp. 10823–10832, IEEE.
- [7] Kevin Wilkinghoff, “Sub-cluster AdaCos: Learning representations for anomalous sound detection,” in *IJCNN*. 2021, IEEE.
- [8] Kevin Wilkinghoff, “An outlier exposed anomalous sound detection system for domain generalization in machine condition monitoring,” Tech. Rep., DCASE Challenge, 2022.
- [9] Aaron E. Rosenberg, Chin-Hui Lee, and Frank K. Soong, “Cepstral channel normalization techniques for HMM-based speaker verification,” in *ICSLP*. 1994, ISCA.
- [10] Kevin Wilkinghoff, “Combining multiple distributions based on sub-cluster AdaCos for anomalous sound detection under domain shifted conditions,” in *DCASE*, 2021, pp. 55–59.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” in *CVPR*. 2016, pp. 770–778, IEEE.
- [12] Lukas Ruff et al., “Deep one-class classification,” in *ICML*. 2018, vol. 80, pp. 4390–4399, PMLR.
- [13] Martín Abadi et al., “Tensorflow: A system for large-scale machine learning,” in *OSDI*, 2016, pp. 265–283.
- [14] Diederik P. Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015.
- [15] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz, “Mixup: Beyond empirical risk minimization,” in *ICLR*, 2018.
- [16] Kota Dohi et al., “MIMII DG: Sound dataset for malfunctioning industrial machine investigation and inspection for domain generalization task,” in *DCASE*. 2022, pp. 31–35, Tampere University.
- [17] Noboru Harada, Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Masahiro Yasuda, and Shoichiro Saito, “ToyADMOS2: Another dataset of miniature-machine operating sounds for anomalous sound detection under domain shift conditions,” in *DCASE*, 2021, pp. 1–5.
- [18] Ying Zeng, Hongqing Liu, Lihua Xu, Yi Zhou, and Lu Gan, “Robust anomaly sound detection framework for machine condition monitoring,” Tech. Rep., DCASE Challenge, 2022.
- [19] Ibuki Kuroyanagi, Tomoki Hayashi, Kazuya Takeda, and Tomoki Toda, “Two-stage anomalous sound detection systems using domain generalization and specialization techniques,” Tech. Rep., DCASE Challenge, 2022.
- [20] Feiyang Xiao et al., “The dcase2022 challenge task 2 system: Anomalous sound detection with self-supervised attribute classification and gmm-based clustering,” Tech. Rep., DCASE Challenge, 2022.
- [21] Yufeng Deng, Jia Liu, and Wei-Qiang Zhang, “Aithu system for unsupervised anomalous detection of machine working status via sounding,” Tech. Rep., DCASE Challenge, 2022.
- [22] Satvik Venkatesh, Gordon Wichern, Aswin Subramanian, and Jonathan Le Roux, “Disentangled surrogate task learning for improved domain generalization in unsupervised anomalous sound detection,” Tech. Rep., DCASE Challenge, 2022.
- [23] Yuming Wei, Jian Guan, Haiyan Lan, and Wenwu Wang, “Anomalous sound detection system with self-challenge and metric evaluation for dcase2022 challenge task 2,” Tech. Rep., DCASE Challenge, 2022.
- [24] Kazuki Morita, Tomohiko Yano, and Khai Tran, “Comparative experiments on spectrogram representation for anomalous sound detection,” Tech. Rep., DCASE Challenge, 2022.
- [25] Jisheng Bai, Yafei Jia, and Siwei Huang, “Jless submission to dcase2022 task2: Batch mixing strategy based method with anomaly detector for anomalous sound detection,” Tech. Rep., DCASE Challenge, 2022.
- [26] Sergey Verbitskiy, Milana Shkhanukova, and Viacheslav Vyshegorodtsev, “Unsupervised anomalous sound detection using multiple time-frequency representations,” Tech. Rep., DCASE Challenge, 2022.

A.1.5 *Key publication 5*

Kevin Wilkinghoff and Fabian Fritz. “On Using Pre-Trained Embeddings for Detecting Anomalous Sounds with Limited Training Data.” In: *31st European Signal Processing Conference*. IEEE, 2023, pp. 186–190. DOI: [10.23919/EUSIPCO58844.2023.10290003](https://doi.org/10.23919/EUSIPCO58844.2023.10290003).

© 2023 IEEE (CC-BY license).

The co-author of this publication contributed in the following ways: *Fabian Fritz* implemented wrapper functions for using the pre-trained embeddings.

On Using Pre-Trained Embeddings for Detecting Anomalous Sounds with Limited Training Data

Kevin Wilkinghoff , Fabian Fritz

Fraunhofer FKIE

Fraunhoferstraße 20, 53343 Wachtberg, Germany
{kevin.wilkinghoff, fabian.fritz}@fkie.fraunhofer.de

Abstract—Using embeddings pre-trained on large datasets as input representations is a popular approach for classifying audio data in case only a few training samples are available. However, for anomalous sound detection pre-trained embeddings usually perform worse than directly training a model because subtle changes indicating anomalous data are not captured sufficiently well. In this paper, the potential of using pre-trained embeddings for detecting anomalous sounds with limited training data is investigated. In experiments conducted on datasets for anomalous sound detection with domain shifts and few-shot open-set classification, it is shown that with increasing openness directly training a model on the original data leads to better performance than using pre-trained embeddings as input. Regardless of the input representation, the presented system achieves a new state-of-the-art performance for few-shot open-set classification in all pre-defined openness settings and is made publicly available.

Index Terms—anomalous sound detection, domain generalization, open-set classification, few-shot learning, transfer learning, representation learning, machine listening

I. INTRODUCTION

Semi-supervised anomalous sound detection (ASD) is the task of identifying anomalous sounds while only having access to normal data when training a system. There are several applications for ASD such as machine condition monitoring for predictive maintenance. Many recent developments have been promoted by the annual DCASE challenge [1]–[3]. For acoustic open-set classification (OSC) problems [4]–[6], normal and anomalous samples have to be distinguished but normal samples also have to be correctly classified as belonging to one of several known classes. Both tasks, ASD and OSC, are especially challenging when only few training samples are available. Examples are few-shot learning [7] for OSC with only k training samples per class (k -shot classification) [5] and ASD under domain shifts [8] between an acoustic source domain with many training samples and a target domain with only very few training samples [2], [3].

One possibility to overcome the difficulties imposed by limited training data is to use embeddings extracted with a neural network pre-trained on other very large datasets [9]. There are several ASD systems [10], [11] based on pre-trained audio embeddings and studies comparing these embeddings for ASD [12] or audio classification tasks [13] in settings with sufficient training data. However, all these systems do not achieve the same state-of-the-art performance as systems directly trained on the data. For acoustic open-set classification, the systems presented in [5], [14] use pre-trained audio

embeddings. Another approach is to use image embeddings for ASD [15] or apply them for zero-shot audio classification [16]. In [17], it is shown that combining multiple hidden representations of pre-trained neural networks improves the performance. Hence, there is a substantial interest in using pre-trained embeddings for classifying audio data. Yet, evaluations in ASD settings with limited training data, which intuitively favor using pre-trained embeddings, are still missing. The goal of this work is to fill this knowledge gap.

The contributions of this work are the following. First and foremost, the ASD performance of using pre-trained audio embeddings, namely VGGish [18], openL3 [19], PANN [20] and Kumar [21] embeddings, are evaluated on the DCASE 2022 ASD dataset with domain shifts [3] and a few-shot OSC dataset [5]. It is shown that only in closed-set classification or with very few unknown classes these embeddings perform better than a model directly trained on the data whereas for more unknown classes the contrary is true even if only very few training samples are available. In conclusion, directly training a model on the data is a better approach for detecting anomalous or unknown samples. Last but not least, the proposed system¹ achieves a new state-of-the-art performance on the few-shot OSC dataset for any of the investigated input representations.

II. PRE-TRAINED AUDIO EMBEDDINGS

For the experimental evaluations in this paper, four different audio embeddings pre-trained on large datasets have been used. These embeddings will now be briefly reviewed.

VGGish [18] is a modified version of the VGG network [22] with a similar architecture. The network is pre-trained in a supervised manner on a preliminary version of YouTube-8M [23], which consists of 2.6 billion audio segments from Youtube videos belonging to a total of 3628 classes. The resulting embeddings have a feature dimension of 128 with an additional time dimension resulting from a sliding window of 960 ms with no overlap applied to the waveforms.

OpenL3 [19] is a network trained to extract *Look, Listen, and Learn (L3)* embeddings [24], [25]. There are multiple versions of the network: One is pre-trained on a music and the other on an environmental subset of AudioSet [26] consisting

¹An open-source implementation of the system is available at: <https://github.com/wilkinghoff/few-shot-open-set-eusipco2023>

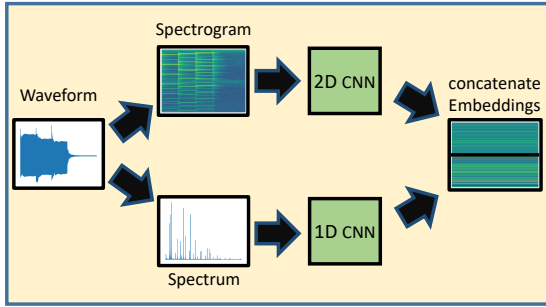


Fig. 1. Structure of the audio embedding model for direct training [28].

of 296K and 195K Youtube videos, respectively. The network is trained in a self-supervised manner to check whether a video frame and an audio clip with a length of one second do or do not belong together using an audio and a video subnetwork. After training, only the audio subnetwork is needed to extract embeddings from audio data. The resulting embeddings have a feature dimension of 512 with an additional time dimension resulting from a sliding window of one second with a hop size of 0.1 seconds applied to the waveforms.

PANN [20] is a combination of a one-dimensional subnetwork applied to waveforms (Wavegram-CNN) and a two-dimensional subnetwork applied to log-mel spectrograms. Both output representations are concatenated and further processed with another two-dimensional subnetwork. The entire network is pre-trained in a supervised manner on AudioSet [26] using a total of 1,934,187 audio clips from Youtube videos belonging to 527 sound classes. As the difference in performance between including and not including Wavegram-CNN is relatively small, we only used the subnetwork with a VGG-like architecture pre-trained on log-mel spectrograms (CNN14). The resulting embeddings have a feature dimension of 2048 with no time dimension because of a global temporal pooling operation inside the network.

Kumar embeddings [21] are extracted using a CNN with a VGG-style architecture [22]. The network is pre-trained in a supervised manner on the balanced subset of AudioSet [26] that consists of around 22,000 audio clips from Youtube videos belonging to 527 sound classes. The resulting embeddings have a feature dimension of 1024 and no time dimension because of a global temporal pooling operation inside the network.

III. SYSTEM DESCRIPTIONS

Different input representations and datasets with different tasks also require different models for further processing. These models will be described in the following subsections. All models are implemented using Tensorflow [27].

A. Anomalous sound detection systems

When directly training a model on the data, i.e. not using pre-trained embeddings, the state-of-the-art ASD system described in [28] is used. This system utilizes two different submodels that use magnitude spectrograms and magnitude

spectra as input representations and are jointly trained to learn embeddings by discriminating among known classes using the sub-cluster AdaCos (scAdaCos) loss [29] with 16 sub-clusters as depicted in Fig. 1. For data augmentation, mixup [30] with a mixing coefficient drawn from a uniform distribution is used. The model is trained for 10 epochs using a batch size of 64 via adam [31]. After training the model, discriminative embeddings are extracted and compared to embeddings belonging to normal training data using cosine similarity (CS). For the source domain, k-means with 16 mean vectors is applied to obtain these normal embeddings and for the target domain, which consists of much fewer samples than the source domain, the embeddings belonging to the training samples themselves are used. Throughout the network, no bias terms and no trainable cluster centers are used as this has been shown to improve the ASD performance. Additional details about the system can be found in [28].

When using pre-trained audio embeddings as input representations, the classification model of the system is replaced with a shallow neural network for transfer learning, whose architecture is very similar to the ones used in [5], [13], [19]. More concretely, the network architecture consists of three hidden layers with dimensions of 512, 128 and 128. Prior to all other operations, all pre-trained embeddings are standardized by using batch normalization as this significantly improves the performance [19]. For the first two hidden layers ReLU is used as an activation function and batch normalization [32] is applied. The last layer does not have an activation function because it serves as the embedding layer. Prior to the last layer dropout [33] with a probability of 50% is applied. Furthermore, mixup [30] with a mixing coefficient drawn from a uniform distribution is applied to the input representations. The network is trained for 100 epochs using a batchsize of 64 by minimizing the scAdaCos loss [29] with 16 sub-clusters using adam [31]. Again, no bias terms or trainable cluster centers are used throughout the network. For the openL3 embeddings, the network pre-trained on the environmental subset has been used.

B. Few-shot open-set classification systems

For OSC, the same systems as used for ASD with only minor modifications are used. This is reasonable because the auxiliary task for training the ASD system is to discriminate among the known classes, which, in addition to detecting anomalies, is exactly the problem to be solved in OSC. All following modifications are used for both models, the model directly trained on the data and the model using pre-trained embeddings as input representations. One modification is using a single sub-cluster for each class because in a few-shot setting only very few training samples are available for each class. Another modification is using a decision threshold for identifying anomalous samples because a threshold-dependent evaluation metric is used for the experiments. A decision is derived as follows. First, the CSs between the embeddings of a test sample and the embeddings of all training samples are computed. Then, the test sample is considered anomalous

if the most similar training sample is one of the known unknown samples or if the highest CS is below a fixed decision threshold. This decision threshold is set to 0.6, 0.65 and 0.75 for an openness of 0 (low), 0.04 (medium) and 0.09 (high), respectively. Following [34], openness is defined as

$$1 - \sqrt{\frac{2 \cdot C_{\text{train}}}{C_{\text{train}} + C_{\text{test}}}} \in [0, 1] \quad (1)$$

where $C_{\text{train}}, C_{\text{test}} \in \mathbb{N}$ denote the total number of classes used for training and testing, respectively. Higher openness values indicate less known or more unknown classes and a value of 0 corresponds to a closed-set classification problem. Furthermore, each model is trained using 100 times the number of shots available for training as the number of epochs and a batch size equal to eight times the number of shots. Using these adaptive hyperparameter settings depending on the size of the training dataset results in a more stable decision threshold across all openness and few-shot settings and thus improves performance. For the openL3 embeddings, the network pre-trained on the music subset has been used.

IV. EXPERIMENTAL RESULTS

A. Datasets

For the experimental evaluations in this paper, two different datasets have been used. The first dataset is the DCASE 2022 ASD dataset [3] for domain generalization in machine condition monitoring. The dataset consists of recordings of machines with real factory background noise each having a length of ten seconds and a sampling rate of 16 kHz, and is split into a training set, a validation set and an evaluation set. The training dataset consists of only normal recordings from seven different machine types: “fan”, “gearbox”, “bearing”, “slide rail”, “valve”, “ToyCar”, and “ToyTrain”. For each machine type, there are six different sections with 990 normal training data samples from the source domain and 10 samples from the target domain with a specific, unknown domain shift meaning that some acoustical characteristics differ for both domains. In addition to these information, there are some attribute information for each training sample that describe the state of machines or noise. Three of the sections belong to the validation set and the other three belong to the evaluation set, each having 100 normal and 100 anomalous samples belonging to the source domain and 100 normal and 100 anomalous samples belonging to the target domain. For none of these test samples, additional information such as attribute information or the domain they belong to are provided. The performance metrics for this dataset are the area under the receiver operating characteristic curve (AUC) and the partial AUC (pAUC) [35], which is the AUC for a low false positive rate ranging from 0 to 0.1 in this case. Both of these metrics are evaluated for each combination of machine type and section regardless of the domain, and the harmonic mean of all derived performance metrics is used as the final result.

The second dataset is a few-shot OSC dataset [5] for acoustic alarm detection in domestic environments. The dataset consists of 24 different alarm sounds and 10 unknown sounds,

TABLE I
HARMONIC MEANS OF AUCs OBTAINED WITH DIFFERENT WAYS TO CONTRACT THE TEMPORAL DIMENSION OF MULTIPLE EMBEDDINGS. FOR PANN AND KUMAR EMBEDDINGS, A SLIDING WINDOW OF 960 MS HAS BEEN APPLIED TO OBTAIN A TIME DIMENSION.

dataset	embeddings	mean of embeddings			mean of scores	native
		before training	during training	after training		
dev set	VGGish	65.78 ± 0.37	64.98 ± 0.25	58.47 ± 0.58	59.27 ± 0.58	not available
dev set	OpenL3	70.94 ± 1.36	70.83 ± 0.93	59.85 ± 0.49	62.67 ± 1.36	not available
dev set	PANN	64.80 ± 0.25	66.30 ± 0.55	59.66 ± 0.32	60.47 ± 0.17	64.21 ± 0.17
dev set	Kumar	66.04 ± 0.76	65.85 ± 0.83	58.94 ± 1.00	62.22 ± 0.98	60.97 ± 0.52
eval set	VGGish	64.69 ± 0.34	63.91 ± 0.73	58.30 ± 0.98	59.78 ± 0.53	not available
eval set	OpenL3	69.06 ± 0.42	68.70 ± 0.94	62.44 ± 0.46	65.02 ± 1.04	not available
eval set	PANN	63.55 ± 0.27	65.29 ± 0.39	58.57 ± 1.07	60.34 ± 0.62	63.33 ± 0.36
eval set	Kumar	63.56 ± 0.59	64.05 ± 0.27	56.95 ± 0.86	61.04 ± 0.44	60.13 ± 0.24

namely “car horn”, “clapping”, “cough”, “door slam”, “engine”, “keyboard tapping”, “music”, “pots and pans”, “steps” and “water falling”. For each of these 34 sound classes, there are 40 different samples with a duration of four seconds and a sampling rate of 16 kHz. There are three different openness [34] settings (“low”, “medium” and “high” where training samples are provided for ten, five or none of the unknown classes, thus corresponding to an openness of 0, 0.04 and 0.09, respectively) and three different numbers of shots (one, two or four) to be used when training the OSC system. For evaluation, the dataset is divided into a different number of validation folds, depending on the number of shots to be used, by using cross-validation. When using one, two or four shots, 40, 20 or 10 validation folds are used, respectively. The performance metric for this dataset is the weighted accuracy with a weight of 0.5, which is the mean of the multiclass accuracy for the known classes and the accuracy for the unknown classes considering only the labels “known” and “unknown”.

Each experiment conducted in this paper is repeated five times and the arithmetic mean and standard deviation are determined as results. Highest values in each row of the tables containing the results are highlighted in bold letters.

B. Anomaly detection in domain-shifted conditions

Some embedding models utilize a sliding window for audio data of arbitrary length and thus consist of multiple embeddings, one for each window position. Therefore, multiple ways of combining pre-trained embeddings belonging to a single recording among the temporal axis are compared first. The results are shown in Tab. I. In contrast to the results obtained in [13], fusing the frame-wise embeddings before or during training leads to significantly better results than fusing the results after training when detecting anomalous data. Furthermore, the fact that using a sliding window for Kumar and PANN embeddings to artificially produce a time dimension improves the performance, shows that temporal structure of the original data needed to detect anomalies is not captured sufficiently well in the pre-trained embeddings.

Next, the following backends for using pre-trained embeddings as input representations are compared: 1) length normalization (LN) and a Gaussian mixture model (GMM), 2) principal component analysis (PCA), LN and a GMM, 3) linear discriminant analysis (LDA), LN and a GMM, 4) a deep neural network (DNN) with categorical cross-entropy (CXE),

LN and a GMM, 5) a DNN with scAdaCos, LN and a GMM, and 6) a DNN with scAdaCos and cosine distance (CD). As shown in Tab II, for both dataset splits and all embedding types, using a shallow classifier as done in [5], [13], [19] significantly improves the performance. Moreover, using the scAdaCos loss function with CS performs best.

Last but not least, the results obtained with pre-trained embeddings are compared to directly training a model on the data as done in [28]. The results can be found in Tab. III. It can be seen that the directly trained model significantly outperforms the shallow classifiers using pre-trained audio embeddings. The most probable reason for this is that the pre-trained embeddings are not designed to and thus do not preserve subtle differences between normal and anomalous samples present in the original data. Another reason is that the recordings are very noisy, which is problematic for the embeddings that have not been exposed to the same noise conditions when being trained on the large datasets (see also [13]). A second observation to be made is that openL3 embeddings perform better than all other pre-trained embeddings, which all have a very similar performance. The most likely reason for this is that these are the only embeddings that are pre-trained in a self-supervised rather than a supervised manner. This is also consistent with the findings in [13].

C. Few-shot open-set classification

The experimental results obtained on the few-shot open-set classification dataset can be found in Tab. IV. The first observation to be made is that regardless of the system, the more shots are available for training and the lower the openness, the higher the mean performance and the smaller the variance gets. This is to be expected because more meaningful training data should always improve the results especially in settings with limited training data. Second, for all openness settings and number of shots all proposed systems outperform both baseline systems presented in [5] by a large margin and thus achieve a new state-of-the-art performance. However, there is a huge difference in performance between different input representations. On average, VGGish embeddings perform worst followed by PANN, OpenL3, Kumar and directly using the data. But interestingly, the best performing input representations have different strengths and weaknesses. OpenL3 embeddings perform best for low openness settings, which is in fact a closed-set classification task. Again, the reason could be that they are obtained by training in a self-supervised rather than a supervised manner. Directly using the data for training performs best in middle or high openness settings, which in contrast to the low openness setting include a semi-supervised ASD subtask. Kumar embeddings have a much higher performance than the system not using any embeddings in a high openness setting when using a single shot per class but perform worse in all other cases. One possible explanation could be the high variance for all performances in this training setting. A last observation to be made is that using pre-trained embeddings tends to be less severely effected when less shots are available for training than when directly using the data to

train the system, which seems reasonable because this is the point of using pre-trained embeddings.

V. CONCLUSIONS

In this work, using pre-trained embeddings for ASD with limited training data has been investigated. In several experiments conducted on the DCASE 2022 ASD dataset and a recently published few-shot OSC dataset, it has been shown that directly training a model leads to better ASD performance than training a shallow classifier with pre-trained audio embeddings. On the OSC dataset, this effect was only evident for middle and high openness settings and the performance gap was not as great as for the ASD dataset. The most likely explanation is that the ASD dataset is very noisy for which pre-trained audio embeddings are known to perform worse whereas the OSC dataset is clean. Moreover, although there are only a few samples for each target domain, there are many training samples belonging to the source domains of the ASD dataset, which seem to provide enough information to also learn meaningful representations of the data in the target domains. The proposed system substantially improves upon the baseline systems of the OSC dataset thus achieves a new state-of-the-art performance and is made publicly available.

REFERENCES

- [1] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, and N. Harada, "Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 81–85.
- [2] Y. Kawaguchi, K. Imoto, Y. Koizumi, N. Harada, D. Niizumi, K. Dohi, R. Tanabe, H. Purohit, and T. Endo, "Description and discussion on dcase 2021 challenge task 2: Unsupervised anomalous detection for machine condition monitoring under domain shifted conditions," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2021, pp. 186–190.
- [3] K. Dohi, K. Imoto, N. Harada, D. Niizumi, Y. Koizumi, T. Nishida, H. Purohit, R. Tanabe, T. Endo, M. Yamamoto, and Y. Kawaguchi, "Description and discussion on DCASE 2022 challenge task 2: Unsupervised anomalous sound detection for machine condition monitoring applying domain generalization techniques," in *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022.
- [4] A. Mesaros, T. Heittola, and T. Virtanen, "Acoustic scene classification in DCASE 2019 challenge: Closed and open set classification and data mismatch setups," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2019, pp. 164–168.
- [5] J. Naranjo-Alcazar, S. Perez-Castanos, P. Zuccarello, A. M. Torres, J. J. Lopez, F. J. Ferri, and M. Cobos, "An open-set recognition and few-shot learning dataset for audio event classification in domestic environments," *Pattern Recognition Letters*, 2022.
- [6] S. Shon, N. Dehak, D. A. Reynolds, and J. R. Glass, "MCE 2018: The 1st multi-target speaker detection and identification challenge evaluation," in *20th Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA, 2019, pp. 356–360.
- [7] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 63:1–63:34, 2021.
- [8] J. Wang, C. Lan, C. Liu, Y. Ouyang, and T. Qin, "Generalizing to unseen domains: A survey on domain generalization," in *Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*. ijcai.org, 2021, pp. 4627–4635.
- [9] T. Reiss, N. Cohen, L. Bergman, and Y. Hoshen, "PANDA: adapting pretrained features for anomaly detection and segmentation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE, 2021, pp. 2806–2814.

TABLE II
HARMONIC MEANS OF AUCs FOR DIFFERENT BACKENDS AND CONSIDERED EMBEDDINGS.

dataset	embedding	LN+GMM	PCA+LN+GMM	LDA+LN+GMM	DNN(CXE) +LN+GMM	DNN(scAdaCos) +LN+GMM	DNN(scAdaCos) +CD
dev set	VGGish	60.22 ± 0.25	60.25 ± 0.43	62.90 ± 0.13	65.40 ± 0.55	64.45 ± 0.60	65.78 ± 0.37
dev set	OpenL3	66.82 ± 0.19	66.33 ± 0.12	64.66 ± 0.24	68.99 ± 0.61	67.83 ± 1.24	70.94 ± 1.36
dev set	PANN	60.36 ± 0.09	61.48 ± 0.18	60.09 ± 0.45	63.82 ± 0.56	64.39 ± 0.75	66.30 ± 0.55
dev set	Kumar	61.47 ± 0.26	61.92 ± 0.29	61.82 ± 0.26	64.08 ± 1.54	64.22 ± 0.63	65.85 ± 0.83
eval set	VGGish	57.48 ± 0.36	57.47 ± 0.18	61.47 ± 0.20	63.77 ± 0.63	62.00 ± 1.26	64.69 ± 0.34
eval set	OpenL3	63.76 ± 0.23	62.62 ± 0.22	64.65 ± 0.33	67.69 ± 0.89	67.18 ± 0.43	69.06 ± 0.42
eval set	PANN	56.18 ± 0.13	60.08 ± 0.08	57.83 ± 1.01	61.63 ± 0.48	63.11 ± 0.48	65.29 ± 0.39
eval set	Kumar	60.00 ± 0.12	60.56 ± 0.13	61.56 ± 0.27	63.42 ± 0.48	61.27 ± 0.30	64.05 ± 0.27

TABLE III
HARMONIC MEANS OF AUCs FOR DIFFERENT INPUT REPRESENTATIONS.

dataset	VGGish	OpenL3	PANN	Kumar	no embedding [28]
dev set	65.78 ± 0.37	70.94 ± 1.36	66.30 ± 0.55	65.85 ± 0.83	81.36 ± 0.66
eval set	64.69 ± 0.34	69.06 ± 0.42	65.29 ± 0.39	64.05 ± 0.27	73.43 ± 0.54

TABLE IV
WEIGHTED ACCURACIES OBTAINED WITH DIFFERENT SYSTEMS AND INPUT REPRESENTATIONS FOR VARIOUS OPENNESS SETTINGS AND NUMBER OF SHOTS PER CLASS. FOR ALL PRE-TRAINED EMBEDDINGS, INDIVIDUAL SETTINGS IDENTIFIED TO PERFORM BEST FOR ASD IN SEC. IV-B ARE USED.

openness	shots	baselines [5]		proposed system using different input representations				
		OpenL3	YAMNet	VGGish	OpenL3	PANN	Kumar	no embedding
low	1	56.8	80.1	90.0 ± 2.2	98.1 ± 1.0	95.6 ± 1.4	96.5 ± 1.3	97.4 ± 1.1
low	2	90.3	88.2	95.6 ± 1.6	99.6 ± 0.3	97.7 ± 0.8	98.5 ± 0.9	99.1 ± 0.7
low	4	97.2	94.9	98.4 ± 0.7	99.9 ± 0.1	98.9 ± 0.5	99.6 ± 0.4	99.7 ± 0.4
middle	1	74.1	78.3	88.7 ± 2.1	97.0 ± 2.3	94.9 ± 1.7	96.1 ± 1.6	96.8 ± 1.4
middle	2	86.7	85.6	93.4 ± 1.8	99.2 ± 0.6	95.7 ± 1.9	97.8 ± 1.3	98.7 ± 0.8
middle	4	91.3	91.9	96.2 ± 1.6	99.3 ± 0.5	97.8 ± 1.1	98.6 ± 1.3	99.8 ± 0.2
high	1	49.9	57.1	84.0 ± 2.6	88.8 ± 5.3	92.1 ± 2.6	94.8 ± 2.4	92.6 ± 4.5
high	2	58.3	61.1	87.8 ± 2.6	94.0 ± 3.2	92.9 ± 2.9	97.0 ± 1.7	97.5 ± 1.7
high	4	60.5	64.3	87.8 ± 2.5	96.1 ± 1.5	96.0 ± 2.1	98.4 ± 1.3	99.1 ± 1.1
arithmetic mean		73.9	77.9	91.3	96.9	95.7	97.5	97.9

[10] S. Grollmisch, D. Johnson, J. Abeßer, and H. Lukashevich, “IAEO3-combining OpenL3 embeddings and interpolation autoencoder for anomalous sound detection,” *Tech. Rep., DCASE2020 Challenge*, 2020.

[11] K. Wilkinghoff, “Using look, listen, and learn embeddings for detecting anomalous sounds in machine condition monitoring,” in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 215–219.

[12] R. Müller, S. Illium, F. Ritz, and K. Schmid, “Analysis of feature representations for anomalous sound detection,” in *13th International Conference on Agents and Artificial Intelligence (ICAART)*. SCITEPRESS, 2021, pp. 97–106.

[13] S. Grollmisch, E. Cano, C. Kehling, and M. Taenzer, “Analyzing the potential of pre-trained embeddings for audio classification tasks,” in *28th European Signal Processing Conference (EUSIPCO)*. IEEE, 2020, pp. 790–794.

[14] K. Wilkinghoff, “On open-set classification with L3-net embeddings for machine listening applications,” in *28th European Signal Processing Conference (EUSIPCO)*. IEEE, 2020, pp. 800–804.

[15] R. Müller, F. Ritz, S. Illium, and C. Linnhoff-Popien, “Acoustic anomaly detection for machine sounds based on image transfer learning,” in *13th International Conference on Agents and Artificial Intelligence (ICAART)*. SCITEPRESS, 2021, pp. 49–56.

[16] D. Dogan, H. Xie, T. Heittola, and T. Virtanen, “Zero-shot audio classification using image embeddings,” in *30th European Signal Processing Conference (EUSIPCO)*. IEEE, 2022, pp. 1–5.

[17] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, “Composing general audio representation by fusing multilayer features of a pre-trained model,” in *30th European Signal Processing Conference (EUSIPCO)*. IEEE, 2022, pp. 200–204.

[18] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. W. Wilson, “CNN architectures for large-scale audio

classification,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 131–135.

[19] A. Cramer, H. Wu, J. Salamon, and J. P. Bello, “Look, listen, and learn more: Design choices for deep audio embeddings,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3852–3856.

[20] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, “PANNs: Large-scale pretrained audio neural networks for audio pattern recognition,” *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 28, pp. 2880–2894, 2020.

[21] A. Kumar, M. Khadkevich, and C. Fügen, “Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 326–330.

[22] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[23] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, “Youtube-8m: A large-scale video classification benchmark,” *CoRR*, vol. abs/1609.08675, 2016. [Online]. Available: <http://arxiv.org/abs/1609.08675>

[24] R. Arandjelovic and A. Zisserman, “Look, listen and learn,” in *International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2017, pp. 609–617.

[25] —, “Objects that sound,” in *15th European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science, vol. 11205. Springer, 2018, pp. 451–466.

[26] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[27] M. Abadi *et al.*, “Tensorflow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[28] K. Wilkinghoff, “Design choices for learning embeddings from auxiliary tasks for domain generalization in anomalous sound detection,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.

[29] —, “Sub-cluster AdaCos: Learning representations for anomalous sound detection,” in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

[30] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “Mixup: Beyond empirical risk minimization,” in *6th International Conference on Learning Representations (ICLR)*, 2018.

[31] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[32] S. Ioffe and C. Szegedy, “Batch normalization: accelerating deep network training by reducing internal covariate shift,” in *32nd International Conference on Machine Learning (ICML)*, vol. 37, 2015, pp. 448–456.

[33] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *CoRR*, vol. abs/1207.0580, 2012.

[34] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, “Toward open set recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 7, pp. 1757–1772, 2013.

[35] D. K. McClish, “Analyzing a portion of the ROC curve,” *Medical decision making*, vol. 9, no. 3, pp. 190–195, 1989.

A.1.6 *Key publication 6*

Kevin Wilkinghoff. “AdaProj: Adaptively Scaled Angular Margin Subspace Projections for Anomalous Sound Detection with Auxiliary Classification Tasks.” Submitted to 9th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE), arXiv:2403.14179. 2024. DOI: [10.48550/arXiv.2403.14179](https://doi.org/10.48550/arXiv.2403.14179).
© 2024 Kevin Wilkinghoff.

AdaProj: Adaptively Scaled Angular Margin Subspace Projections for Anomalous Sound Detection with Auxiliary Classification Tasks

Kevin Wilkinghoff

Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE

Fraunhoferstraße 20, 53343 Wachtberg, Germany

kevin.wilkinghoff@ieee.org

Abstract—One of the state-of-the-art approaches for semi-supervised anomaly detection is to first learn an embedding space and then estimate the distribution of normal data. This can be done by using one-class losses or by using auxiliary classification tasks based on meta information or self-supervised learning. Angular margin losses are a popular training objective because they increase intra-class similarity and avoid learning trivial solutions by reducing inter-class similarity. In this work, AdaProj a novel loss function that generalizes upon angular margin losses is presented. In contrast to angular margin losses, which project data of each class as close as possible to their corresponding class centers, AdaProj learns to project data onto class-specific subspaces. By doing so, the resulting distributions of embeddings belonging to normal data are not required to be as restrictive as other loss functions allowing a more detailed view on the data. This enables a system to more accurately detect anomalous samples during testing. In experiments conducted on the DCASE2022 and DCASE2023 datasets, it is shown that using AdaProj to learn an embedding space significantly outperforms other commonly used loss functions achieving a new state-of-the-art performance on the DCASE2023 dataset.

Index Terms—machine listening, anomaly detection, representation learning, angular margin loss, domain generalization

I. INTRODUCTION

Semi-supervised anomaly detection is the task of training a system to differentiate between normal and anomalous data using only normal training samples [1]. For many applications, it is much less costly to collect normal data than anomalous samples for training a system because anomalies occur only rarely and intentionally generating them is often costly. Therefore, for these applications a semi-supervised anomaly detection setting is often a realistic assumption. An example application is acoustic machine condition monitoring for predictive maintenance, which is largely promoted through the anomalous sound detection (ASD) tasks of the annual DCASE Challenge [2]–[5] and will serve as the running example in this work. Here, normal data corresponds to sounds of fully functioning machines whereas anomalous sounds indicate mechanical failure. The goal is to develop a system that detects anomalous recordings using only normal recordings as training data.

One of the main difficulties to overcome is that it is practically impossible to record isolated sounds of a target machine. Instead, recordings also contain many other sounds

emitted by non-target machines or humans. Compared to this complex acoustic scene, anomalous signal components of the target machines are very subtle and hard to detect without utilizing additional knowledge. Another main difficulty is that a system should also be able to reliably detect anomalous sounds when changing the acoustic conditions or machine settings without needing to collect large amounts of data in this changed conditions or to re-train the system (domain generalization [6]). One possibility to simultaneously overcome both difficulties is to learn mapping audio signals into a fixed-dimensional vector space, in which representations belonging to normal and anomalous data, called embeddings, can be easily separated. Then, by estimating the distribution in the embedding space of normal training samples, one can compute an anomaly score for an unseen test sample by computing the likelihood of this sample being normal or simply measuring the distance to normal samples. To train such an embedding model, the state-of-the-art is to utilize an auxiliary classification task using provided meta information or self-supervised learning. This enables the embedding model to closely monitor target signals and ignore other signals and noise [7]. For machine condition monitoring, possible auxiliary tasks are classifying between machine types [8]–[10] or, additionally, between different machine states and noise settings [11]–[13], recognizing augmented and non-augmented versions of normal data [8], [14] or predicting the activity of machines [12]. Using an auxiliary task to learn embeddings is also called outlier exposure (OE) [15] because normal samples belonging to other classes than a target class can be considered proxy outliers [16].

The contributions of this work are the following. First and foremost, AdaProj, a novel angular margin loss function that learns class-specific subspaces for training an embedding model, is presented. Second, it is proven that AdaProj has arbitrarily large optimal solution spaces allowing to relax the compactness requirements of the class-specific distributions in the embedding space. Last but not least, AdaProj is compared to other commonly used loss functions. In experiments conducted on the DCASE2022 and DCASE2023 ASD datasets it is shown that AdaProj outperforms all other loss functions. As a result, a new state-of-the-art performance is achieved on the

A. Related Work

When training a neural network to solve a classification task, usually the softmax function in combination with the categorical cross-entropy (CCE) is used. However, directly training a network this way only reduces inter-class similarity without explicitly increasing intra-class similarity [17]. When training an embedding model for anomaly detection, high intra-class similarity is a desired property to cluster normal data and be able to detect anomalous samples. To address this issue, losses should also explicitly increase intra-class similarity.

There are several loss functions to achieve this. [18] proposed a compactness loss to project the data into a hypersphere of minimal volume for one-class classification. However, for machine condition monitoring in noisy conditions it is known that one-class losses perform worse than losses that also discriminatively solve an auxiliary classification task [7]. [19] did this by simultaneously using a so-called descriptiveness loss consisting of a CCE on another arbitrary dataset than the target dataset, to be able to learn a better structured embedding space in case no meta information are available on the target dataset. For machine condition monitoring, often meta information is available as it can at least be ensured which machine is being recorded when collecting data. [10] used center loss [20], which minimizes the distance to learned class centers for each class. Another choice are angular margin losses that learn an embedding space on the unit sphere while ensuring a margin between classes leading to better generalization capabilities than losses that utilize the whole Euclidean space. Specific examples are the additive margin softmax loss [17] as used by [9], [21] and ArcFace [22] as used by [8], [13], [23]. [24], [25] use the AdaCos loss [26], which essentially is ArcFace with an adaptive scale parameter, or the sub-cluster AdaCos loss [27], which utilizes multiple sub-clusters instead of a single one.

As stated above, the goal of this work is to further extend these loss functions to learning class-specific linear subspaces to relax the compactness requirements and allow more flexibility for the network when learning to map audio data into an embedding space. There are also other works utilizing losses to learn subspaces based on orthogonal projections to learn embedding spaces for other applications in different ways. [28] used orthogonal projections as a constraint for training an autoencoder based anomaly detection system. Another example is semi-supervised image classification by using a combination of class-specific subspace projections with a reconstructions loss and ensure that they are different by also using a discriminative loss [29]. Our work focuses on learning an embedding space through an auxiliary classification task that is well-suited for semi-supervised anomaly detection.

¹The source code will be made available after the review process to not reveal the identity of the authors.

A. Notation

Let $\phi : X \rightarrow \mathbb{R}^D$ denote a neural network where X denotes some input space, which consists of audio signals in this work, and $D \in \mathbb{N}$ denotes the dimension of the embedding space. Define the linear projection of $x \in \mathbb{R}^D$ onto the subspace $\text{span}(\mathcal{C}_k) \subset \mathbb{R}^D$ as $P_{\text{span}(\mathcal{C}_k)}(x) := \sum_{c_k \in \mathcal{C}_k} \langle x, c_k \rangle c_k$. Furthermore, let $\mathcal{S}^{D-1} = \{y \in \mathbb{R}^D : \|y\|_2 = 1\} \subset \mathbb{R}^D$ denote the D -sphere and define $P_{\mathcal{S}^{D-1}}(x) := \frac{x}{\|x\|_2} \in \mathcal{S}^{D-1}$ to be the projection onto the D -sphere.

B. AdaProj loss function

Similar to the sub-cluster AdaCos loss [27], the idea of the AdaProj loss is to enlarge the space of optimal solutions to allow the network to learn less restrictive distributions of the normal samples. This may help to differentiate between normal and anomalous data after training. The reason is that for some auxiliary classes a strong compactness may be detrimental when aiming to learn an embedding space that separates normal and anomalous data since both may be mapped onto the same compact distribution making it impossible to distinguish them. This relaxation is achieved by measuring the distance to class-specific subspaces while training the embedding model instead of measuring the distance to a single or multiple centers as done for other angular margin losses.

Formally, the definition of the AdaProj loss is as follows.

Definition 1 (AdaProj loss). Let $\mathcal{C}_k \subset \mathbb{R}^D$ with $|\mathcal{C}_k| = J \in \mathbb{N}$ denote class centers for class $k \in \{1, \dots, N_{\text{classes}}\}$. Then for the AdaProj loss the logit for class $k \in \{1, \dots, N_{\text{classes}}\}$ is defined as

$$L(x, \mathcal{C}_k) := \hat{s} \cdot \|P_{\mathcal{S}^{D-1}}(x) - P_{\mathcal{S}^{D-1}}(P_{\text{span}(\mathcal{C}_k)}(x))\|_2^2$$

where $\hat{s} \in \mathbb{R}_+$ is the so-called dynamically adaptive scale parameter of the AdaCos loss [26]. Inserting these logits into a softmax function and computing the CCE yields the AdaProj loss function.

Remark. Note that, by Lemma 5 of [7], it holds that

$$\begin{aligned} & \|P_{\mathcal{S}^{D-1}}(x) - P_{\mathcal{S}^{D-1}}(P_{\text{span}(\mathcal{C}_k)}(x))\|_2^2 \\ &= 2(1 - \langle P_{\mathcal{S}^{D-1}}(x), P_{\mathcal{S}^{D-1}}(P_{\text{span}(\mathcal{C}_k)}(x)) \rangle), \end{aligned}$$

which is equal to the cosine distance in this case. This explains why the AdaProj loss can be called an angular margin loss.

As for other angular margin losses, projecting the embedding space onto the D -sphere has several advantages [7]. Most importantly, if D is sufficiently large randomly initialized centers are with very high probability approximately orthonormal to each other [30], i.e. distributed equidistantly, and sufficiently far away from $\mathbf{0} \in \mathbb{R}^D$. Therefore, one does not need to carefully design a method to initialize the centers. Another advantage is that a normalization may prevent numerical issues, similar to applying batch normalization [31].

The following Lemma shows that using the AdaProj loss, as defined above, indeed allows the network to utilize a larger solution space.

Lemma 2. Let $x \in \mathbb{R}^D$ and let $\mathcal{C} \subset \mathbb{R}^D$ contain pairwise orthonormal elements. If $x \in \text{span}(\mathcal{C}) \cap \mathcal{S}^{D-1}$, then

$$\|P_{\mathcal{S}^{D-1}}(x) - P_{\mathcal{S}^{D-1}}(P_{\text{span}(\mathcal{C})}(x))\|_2^2 = 0.$$

Proof. Let $x \in \text{span}(\mathcal{C}) \cap \mathcal{S}^{D-1} \subset \mathbb{R}^D$ with $|\mathcal{C}| = J$. Therefore, $\|x\|_2 = 1$ and there are $\lambda_j \in \mathbb{R}$ with $x = \sum_{j=1}^J \lambda_j c_j$. Thus, it holds that

$$\begin{aligned} x &= \sum_{j=1}^J \lambda_j c_j = \sum_{j=1}^J \sum_{i=1}^J \lambda_i \langle c_i, c_j \rangle c_j = \sum_{j=1}^J \langle \sum_{i=1}^J \lambda_i c_i, c_j \rangle c_j \\ &= \sum_{j=1}^J \langle x, c_j \rangle c_j = P_{\text{span}(\mathcal{C})}(x). \end{aligned}$$

Hence, we obtain

$$\|P_{\mathcal{S}^{D-1}}(x) - P_{\mathcal{S}^{D-1}}(P_{\text{span}(\mathcal{C})}(x))\|_2^2 = 0. \quad \square$$

Remark. If \mathcal{C} contains randomly initialized elements of the unit sphere and D is sufficiently large, then the elements of \mathcal{C} are approximately pairwise orthonormal with very high probability [30]. Hence, this Lemma is likely to hold for the AdaProj loss.

When inserting the projection onto the $D - 1$ -sphere as an operation into the neural network, this Lemma shows that the solution space for the AdaProj loss function is increased to the whole subspace $\text{span}(\mathcal{C})$, which has a dimension of $|\mathcal{C}|$ with very high probability. Because of this, it should be ensured that $|\mathcal{C}| < D$. Otherwise the whole embedding space may be an optimal solution and thus the network cannot learn a meaningful embedding space. In comparison, for the AdaCos loss only the class centers themselves are optimal solutions and for the sub-cluster AdaCos loss each sub-cluster is an optimal solution [7].

III. EXPERIMENTAL RESULTS

To experimentally evaluate the proposed AdaProj loss, first the experimental setup is presented by describing the datasets and the ASD system. After that, experimental results regarding the performance obtained with different loss functions, the impact of the subspace dimension and the relation to state-of-the-art systems are presented and discussed.

A. Datasets and performance metrics

For the experiments, two ASD datasets, namely the DCASE2022 ASD dataset [4] and the DCASE2023 ASD dataset [5] for semi-supervised machine condition monitoring, were used. Both datasets consist of a development set and an evaluation set that are divided into a training split containing only normal data and a test split containing normal as well as anomalous data. Furthermore, both tasks explicitly capture the problem of domain generalization [6] by defining a source and a target domain, which differs from the source domain by altering machine parameters or noise conditions. The task is to detect anomalous samples regardless of the domain a sample belongs to by training a system with only normal data. As meta information, the target machine type of each

sample is known and for the training samples, also the domain and additional parameter settings or noise conditions, called attribute information, are known and thus can be utilized to train an embedding model.

The DCASE2022 ASD dataset [4] consists of the machine types “ToyCar” and “ToyTrain” from ToyAdmos2 [32] and “fan”, “gearbox”, “bearing”, “slide rail” and “valve” from MIMII-DG [33]. For each machine type, there are six different so-called sections, indicating different machine IDs of these types of which three belong to the development set and three belong to the test set. These IDs are known for each recording and can also be utilized as meta information to train the system. For the source domain of each section, there are 1000 normal audio recordings with a duration of 10s with a sampling rate of 16 kHz belonging to the training split and 50 normal and 50 anomalous samples belonging to the test split. For the target domain of each section, there are also approximately 50 normal and 50 anomalous samples belonging to the test split but only 10 normal audio recordings belonging to the training split.

The DCASE2023 ASD dataset [5] is similar to the DCASE2022 ASD dataset with the following modifications. First of all, the development set and the evaluation set contain mutually exclusive machine types. More concretely, the development set contains the same machine types as the DCASE2022 dataset and the evaluation set contains the machine types “ToyTank”, “ToyNscale” and “ToyDrone” from ToyAdmos2+ [34] and “vacuum”, “bandsaw”, “grinder” and “shaker” from MIMII-DG [33]. Furthermore, there is only a single section for each machine type, which makes the auxiliary classification task much easier resulting in less informative embeddings for the ASD task. Last but not least, the duration of each recording has a length between 6s and 18s. Overall, all three modifications make this task much more challenging than the DCASE2022 ASD task.

To measure the performance of the ASD systems the threshold-independent area under the receiver operating characteristic (ROC) curve (AUC) metric is used. In addition, the partial area under the ROC curve (pAUC) [35], which is the AUC for low false positive rates ranging from 0 to $p = 0.1$ in this case, is used. The reason for incorporating the pAUC is that, for machine condition monitoring, one is interested in ensuring a low false alarm rate to not lose the trust of users in taking alarms seriously. Both performance metrics are computed domain-independent for every previously defined section of the dataset and the harmonic mean of all resulting values is the final score used to measure and compare the performances of different ASD systems.

B. Anomalous sound detection system

For all experiments conducted in this work, the state-of-the-art ASD system presented in [25] is used. An overview of the system can be found in Figure 1. The system consists of three main components: 1) a feature extractor, 2) an embedding model and 3) a backend for computing anomaly scores.

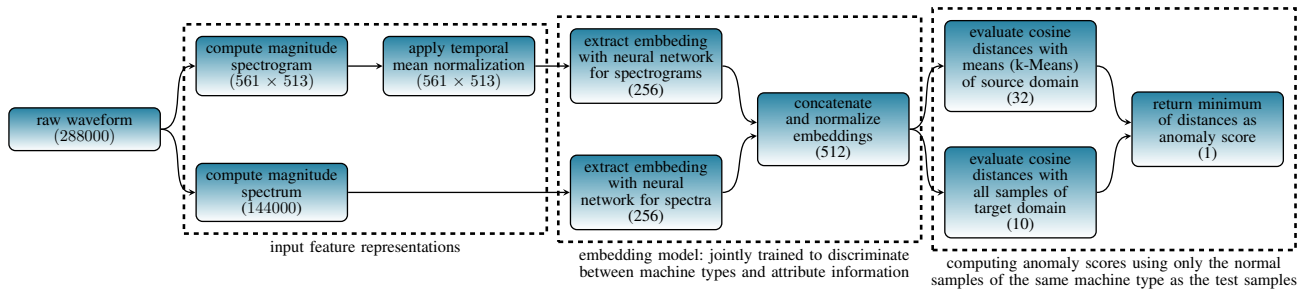


Fig. 1. Structure of the ASD system, adapted from Figure 1 in [36]. Representation size in each step is given in brackets.

In the first processing block, two different feature representations are extracted from the raw waveforms, namely magnitude spectrograms and the full magnitude spectrum. To make both feature representations a bit more different the temporal mean is subtracted from the magnitude spectrograms, essentially removing static frequency information that are captured with the highest possible resolution in the magnitude spectrums. Utilizing both of these representations was shown to significantly improve the performance [25] despite their close relation.

For each of the two feature representations, another convolutional subnetwork is trained and the resulting embeddings are concatenated and normalized with respect to the Euclidean norm to obtain a single embedding. In contrast to the original architecture, the embedding dimension is doubled from 256 to 512. More details about the subnetwork architectures can be found in [25]. The network is trained for 10 epochs using a batch size of 64 using adam [37] by utilizing meta information such as machine types and the provided attribute information as an auxiliary classification task. Different loss functions can be used for this purpose and will be compared in the next subsection. All loss functions investigated in this work require class-specific center vectors, which are initialized randomly using Glorot uniform initialization [38]. To improve the resulting ASD performance, the randomly initialized class centers are not adapted during training and no bias terms are used as proposed in [18] for deep one-class classification. Furthermore, mixup [39] with a uniformly distributed mixing coefficient is applied to the waveforms.

As a backend, k-means with 32 means is applied to the normal training samples of the source domain. For a given test sample, the smallest cosine distance to these means and the ten normal training samples of the target domain is used as an anomaly score. Thus, smaller values indicate normal samples whereas higher values indicate anomalous samples.

C. Performance evaluation

The first and most important experiment is to compare different loss functions for training the embedding extractor of the ASD system presented in the previous subsection. For this purpose, we used 1) individual class-specific IC compactness losses jointly trained on all classes, as proposed for one-class classification in [18], 2) an additional discriminative

CCE loss, similar to the descriptiveness loss used in [19] but trained on the same dataset, 3) the AdaCos loss [26], 4) the sub-cluster AdaCos loss [27] with 32 sub-clusters and 5) the proposed AdaProj loss. The experiments were conducted on the development and evaluation split of the DCASE2022 and the DCASE2023 ASD dataset and each experiment was repeated ten times to reduce the variance of the resulting performances. Furthermore, the arithmetic means of the performances obtained on the different datasets are shown to be able to directly compare the overall performance. The results can be found in Table I.

The main observation to be made is that the proposed AdaProj loss clearly outperforms all other losses on both datasets. Especially on the DCASE2023 dataset, there are significant improvements to be observed. The most likely explanation is that for this dataset the classification task is less difficult and thus a few classes may be easily identified leading to embeddings that do not carry enough information to distinguish between embeddings belonging to normal and anomalous samples of these classes.

Another interesting observation is that, in contrast to the original results presented in [27], the sub-cluster AdaCos loss actually performs slightly worse than the AdaCos loss despite having a higher solution space. A possible explanation is that in [27], the centers are adapted during training whereas, in our work, they are not as this has been shown to improve the resulting performance [25]. Since all centers have approximately the same distance to each other when being randomly initialized, i.e. the centers belonging to a target class and the other centers, the network will likely utilize only a single center for each class that is closest to the initial embeddings of the corresponding target class. Moreover, a low inter-class similarity is more difficult to ensure due to the higher total number of sub-clusters belonging to other classes. This leads to more restrictive requirements when learning class-specific distributions and thus actually reduces the ability to differentiate between embeddings belonging to normal and anomalous samples.

D. Investigating the impact of the subspace dimension on the performance

As an ablation study, different choices for the dimension of the subspaces have been compared experimentally on the

TABLE I

ASD PERFORMANCE OBTAINED WITH DIFFERENT LOSS FUNCTIONS. HARMONIC MEANS OF ALL AUCs AND PAUCs OVER ALL PRE-DEFINED SECTIONS OF THE DATASET ARE DEPICTED IN PERCENT. ARITHMETIC MEAN AND STANDARD DEVIATION OF THE RESULTS OVER TEN INDEPENDENT TRIALS ARE SHOWN. BEST RESULTS IN EACH COLUMN ARE HIGHLIGHTED WITH BOLD LETTERS.

DCASE2022 development set [4]						
loss function	source domain		target domain		domain-independent	
	AUC	pAUC	AUC	pAUC	AUC	pAUC
intra-class (IC) compactness loss [18]	81.8 ± 1.6	74.9 ± 1.7	75.3 ± 1.0	63.4 ± 0.6	79.2 ± 0.9	64.7 ± 1.1
IC compactness loss + CCE [19]	82.5 ± 1.8	75.5 ± 0.9	75.5 ± 0.7	61.6 ± 0.9	79.0 ± 0.8	65.0 ± 0.7
AdaCos loss [26]	82.6 ± 1.4	76.0 ± 1.1	76.5 ± 1.2	62.3 ± 1.4	79.8 ± 0.7	65.5 ± 0.9
sub-cluster AdaCos loss [27]	83.2 ± 2.1	75.9 ± 1.3	77.6 ± 1.0	62.1 ± 1.5	80.0 ± 1.4	65.2 ± 1.1
proposed AdaProj loss	84.3 ± 1.1	76.3 ± 1.1	77.2 ± 1.2	62.2 ± 1.1	80.6 ± 0.8	65.5 ± 1.3
DCASE2022 evaluation set [4]						
loss function	source domain		target domain		domain-independent	
	AUC	pAUC	AUC	pAUC	AUC	pAUC
IC compactness loss [18]	74.7 ± 0.9	64.2 ± 1.3	65.9 ± 0.8	57.8 ± 0.9	70.3 ± 0.8	58.9 ± 0.8
IC compactness loss + CCE [19]	75.6 ± 0.7	66.9 ± 0.8	69.3 ± 0.7	59.3 ± 0.7	72.6 ± 0.4	60.3 ± 0.7
AdaCos loss [26]	77.2 ± 0.5	65.9 ± 1.4	68.6 ± 1.1	58.6 ± 0.7	73.0 ± 0.4	59.7 ± 0.6
sub-cluster AdaCos loss [27]	77.0 ± 0.7	66.5 ± 0.9	68.3 ± 0.8	58.8 ± 0.6	72.9 ± 0.6	59.5 ± 0.5
proposed AdaProj loss	77.4 ± 1.0	67.0 ± 0.6	69.7 ± 0.6	59.6 ± 0.6	73.6 ± 0.7	60.5 ± 0.7
DCASE2023 development set [5]						
loss	source domain		target domain		domain-independent	
	AUC	pAUC	AUC	pAUC	AUC	pAUC
IC compactness loss [18]	67.0 ± 2.1	62.4 ± 1.0	69.1 ± 1.4	56.4 ± 1.1	67.7 ± 1.2	56.9 ± 0.9
IC compactness loss + CCE [19]	70.6 ± 1.8	64.1 ± 1.8	71.2 ± 1.4	55.5 ± 1.6	70.4 ± 1.0	57.4 ± 1.1
AdaCos loss [26]	70.7 ± 1.3	64.3 ± 1.1	71.2 ± 1.1	55.4 ± 1.3	70.9 ± 0.9	56.8 ± 0.9
sub-cluster AdaCos loss [27]	68.3 ± 1.7	62.0 ± 1.5	71.8 ± 1.5	55.6 ± 1.5	70.4 ± 0.9	56.3 ± 0.8
proposed AdaProj loss	70.3 ± 1.7	61.8 ± 1.6	72.2 ± 1.4	55.1 ± 1.1	71.4 ± 1.0	56.2 ± 0.7
DCASE2023 evaluation set [5]						
loss	source domain		target domain		domain-independent	
	AUC	pAUC	AUC	pAUC	AUC	pAUC
IC compactness loss [18]	73.5 ± 1.8	63.4 ± 1.8	58.8 ± 2.5	55.7 ± 1.3	64.0 ± 1.5	55.8 ± 0.9
IC compactness loss + CCE [19]	74.3 ± 1.5	64.0 ± 1.6	61.6 ± 2.0	55.7 ± 0.9	67.5 ± 0.8	57.5 ± 1.0
AdaCos loss [26]	74.7 ± 1.5	63.8 ± 1.8	61.6 ± 3.4	57.1 ± 1.4	68.0 ± 1.6	58.0 ± 1.1
sub-cluster AdaCos loss [27]	73.2 ± 1.9	61.6 ± 1.4	62.0 ± 2.2	55.8 ± 1.3	66.5 ± 1.6	56.2 ± 1.0
proposed AdaProj loss	74.2 ± 1.8	62.9 ± 1.0	64.4 ± 2.0	57.7 ± 0.8	69.8 ± 1.3	60.0 ± 0.5
arithmetic mean over all datasets						
loss	source domain		target domain		domain-independent	
	AUC	pAUC	AUC	pAUC	AUC	pAUC
IC compactness loss [18]	74.3	66.2	67.3	58.3	70.3	59.1
IC compactness loss + CCE [19]	75.8	67.6	69.4	58.0	72.4	60.1
AdaCos loss [26]	76.3	67.5	69.5	58.4	72.9	60.0
sub-cluster AdaCos loss [27]	75.4	66.5	69.9	58.1	72.5	59.3
proposed AdaProj loss	76.6	66.3	70.9	58.7	73.9	60.6

DCASE2023 ASD dataset. The results can be found in Figure 2. It can be seen, that, on the development set, the results are relatively stable while a larger dimension slightly improves the performance on the evaluation set without any significant differences. For subspace dimensions greater than 48 the performances seem to slightly degrade again. In conclusion, the subspace dimension should be neither too high nor too low and a dimension of 32 as used for the other experiments in this works appears to be a reasonable choice.

E. Comparison to other published systems

As a last experiment, the performance of the proposed system using AdaProj is compared to the ten top-performing systems of the DCASE2023 Challenge. As many systems

utilize ensembles of models, the mean of the anomaly scores belonging to ten independent trials was used to create an ensemble of ten systems allowing a fair comparison. The results can be found in Figure 3. It can be seen that the proposed system outperforms all other published systems and thus achieves a new state-of-the-art performance. This adds confidence to the benefits of the AdaProj loss function.

IV. CONCLUSIONS AND FUTURE WORK

In this work, AdaProj a novel angular margin loss function specifically designed for semi-supervised anomaly detection with auxiliary classification tasks was presented. It was proven that this loss function learns an embedding space with class-specific subspaces of arbitrary dimension. In contrast to other

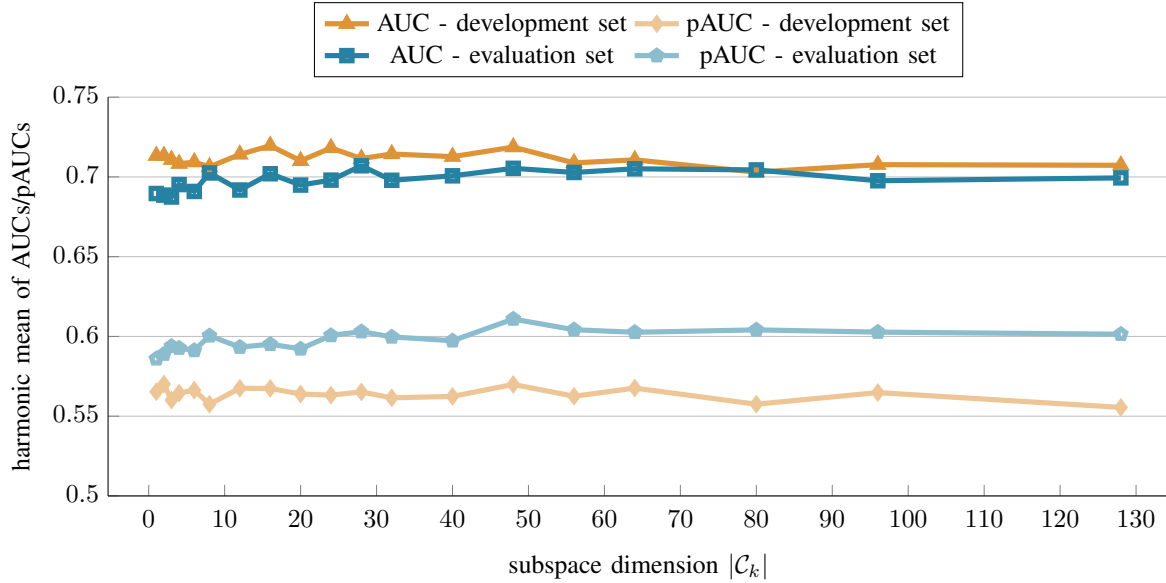


Fig. 2. Domain-independent performance obtained on the DCASE2023 dataset with different subspace dimensions. The means over ten independent trials are shown.

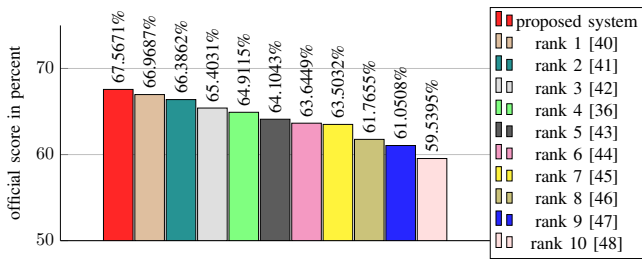


Fig. 3. Comparison of the proposed system to the ten top-performing systems of the DCASE2023 Challenge. The official evaluation script was used to compute the score.

angular margin losses, which try to project data to individual points in space, this relaxes the requirements of solving the classification task and allows for less compact distributions in the embedding space. In experiments conducted on the DCASE2022 and DCASE2023 ASD datasets, it was shown that using AdaProj results in better performance than other commonly used loss functions. In conclusion, the resulting embedding space has a more desirable structure than the other embedding spaces for differentiating between normal and anomalous samples. As a result, a new state-of-the-art performance outperforming all other published systems could be achieved on the DCASE2023 ASD dataset. For future work, it is planned to evaluate AdaProj on other datasets and using other auxiliary classification tasks, e.g. tasks imposed by self-supervised learning.

REFERENCES

- [1] C. Aggarwal, *Outlier Analysis*, 2nd ed. Springer, 2017.
- [2] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, and N. Harada, "Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *5th Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 81–85.
- [3] Y. Kawaguchi, K. Imoto, Y. Koizumi, N. Harada, D. Niizumi, K. Dohi, R. Tanabe, H. Purohit, and T. Endo, "Description and discussion on DCASE 2021 challenge task 2: Unsupervised anomalous detection for machine condition monitoring under domain shifted conditions," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2021, pp. 186–190.
- [4] K. Dohi, K. Imoto, N. Harada, D. Niizumi, Y. Koizumi, T. Nishida, H. Purohit, R. Tanabe, T. Endo, M. Yamamoto, and Y. Kawaguchi, "Description and discussion on DCASE 2022 challenge task 2: Unsupervised anomalous sound detection for machine condition monitoring applying domain generalization techniques," in *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022, pp. 26–30.
- [5] K. Dohi, K. Imoto, N. Harada, D. Niizumi, Y. Koizumi, T. Nishida, H. Purohit, R. Tanabe, T. Endo, and Y. Kawaguchi, "Description and discussion on DCASE 2023 challenge task 2: First-shot unsupervised anomalous sound detection for machine condition monitoring," in *8th Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. Tampere University, 2023, pp. 31–35.
- [6] J. Wang, C. Lan, C. Liu, Y. Ouyang, and T. Qin, "Generalizing to unseen domains: A survey on domain generalization," in *Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*. ijcai.org, 2021, pp. 4627–4635.
- [7] K. Wilkinghoff and F. Kurth, "Why do angular margin losses work well for semi-supervised anomalous sound detection?" 2023, arXiv:2309.15643.
- [8] R. Giri, S. V. Tenneti, F. Cheng, K. Helwani, U. Isik, and A. Krishnaswamy, "Self-supervised classification for detecting anomalous sounds," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 46–50.
- [9] J. A. Lopez, H. Lu, P. Lopez-Meyer, L. Nachman, G. Stemmer, and J. Huang, "A speaker recognition approach to anomaly detection," in *5th Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 96–99.
- [10] T. Inoue, P. Vinayavekhin, S. Morikuni, S. Wang, T. Hoang Trong, D. Wood, M. Tatsubori, and R. Tachibana, "Detection of anomalous sounds for machine condition monitoring using classification confidence," in *5th Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 66–70.
- [11] S. Venkatesh, G. Wichern, A. S. Subramanian, and J. L. Roux, "Im-

- proved domain generalization via disentangled multi-task learning in unsupervised anomalous sound detection,” in *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022, pp. 196–200.
- [12] T. Nishida, K. Dohi, T. Endo, M. Yamamoto, and Y. Kawaguchi, “Anomalous sound detection based on machine activity detection,” in *30th European Signal Processing Conference EUSIPCO*. IEEE, 2022, pp. 269–273.
- [13] Y. Deng, A. Jiang, Y. Duan, J. Ma, X. Chen, J. Liu, P. Fan, C. Lu, and W. Zhang, “Ensemble of multiple anomalous sound detectors,” in *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022.
- [14] H. Chen, Y. Song, Z. Zhuo, Y. Zhou, Y.-H. Li, H. Xue, and I. McLoughlin, “An effective anomalous sound detection method based on representation learning with simulated anomalies,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.
- [15] D. Hendrycks, M. Mazeika, and T. G. Dietterich, “Deep anomaly detection with outlier exposure,” in *7th International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2019.
- [16] P. Primus, V. Haunschmid, P. Praher, and G. Widmer, “Anomalous sound detection as a simple binary classification problem with careful selection of proxy outlier examples,” in *5th Workshop on Detection and Classification of Acoustic Scenes and Events 2020 (DCASE)*, 2020, pp. 170–174.
- [17] F. Wang, J. Cheng, W. Liu, and H. Liu, “Additive margin softmax for face verification,” *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.
- [18] L. Ruff, N. Görnitz, L. Deecke, S. A. Siddiqui, R. A. Vandermeulen, A. Binder, E. Müller, and M. Kloft, “Deep one-class classification,” in *35th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 4390–4399.
- [19] P. Perera and V. M. Patel, “Learning deep features for one-class classification,” *IEEE Transactions on Image Processing*, vol. 28, no. 11, pp. 5450–5463, 2019.
- [20] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, “A discriminative feature learning approach for deep face recognition,” in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 499–515.
- [21] J. A. Lopez, G. Stemmer, P. Lopez-Meyer, P. Singh, J. A. del Hoyo Ontiveros, and H. A. Cordourier, “Ensemble of complementary anomaly detectors under domain shifted conditions,” in *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2021, pp. 11–15.
- [22] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “ArcFace: Additive angular margin loss for deep face recognition,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 4690–4699.
- [23] I. Kuroyanagi, T. Hayashi, Y. Adachi, T. Yoshimura, K. Takeda, and T. Toda, “An ensemble approach to anomalous sound detection based on conformer-based autoencoder and binary classifier incorporated with metric learning,” in *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2021, pp. 110–114.
- [24] K. Wilkinghoff, “Combining multiple distributions based on sub-cluster adacos for anomalous sound detection under domain shifted conditions,” in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2021, pp. 55–59.
- [25] —, “Design choices for learning embeddings from auxiliary tasks for domain generalization in anomalous sound detection,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.
- [26] X. Zhang, R. Zhao, Y. Qiao, X. Wang, and H. Li, “AdaCos: Adaptively scaling cosine logits for effectively learning deep face representations,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 10 823–10 832.
- [27] K. Wilkinghoff, “Sub-cluster AdaCos: Learning representations for anomalous sound detection,” in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.
- [28] Q. Yu, M. S. Kavitha, and T. Kurita, “Autoencoder framework based on orthogonal projection constraints improves anomalies detection,” *Neurocomputing*, vol. 450, pp. 372–388, 2021.
- [29] L. Li, Y. Zhang, and A. Huang, “Learnable subspace orthogonal projection for semi-supervised image classification,” in *16th Asian Conference on Computer Vision (ACCV)*, ser. Lecture Notes in Computer Science, vol. 13843. Springer, 2022, pp. 477–490.
- [30] A. N. Gorban, I. Y. Tyukin, D. V. Prokhorov, and K. I. Sofeikov, “Approximation with random bases: Pro et contra,” *Information Sciences*, vol. 364–365, pp. 129–145, 2016.
- [31] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *32nd International Conference on Machine Learning (ICML)*, vol. 37, 2015, pp. 448–456.
- [32] N. Harada, D. Niizumi, D. Takeuchi, Y. Ohishi, M. Yasuda, and S. Saito, “ToyADMOS2: Another dataset of miniature-machine operating sounds for anomalous sound detection under domain shift conditions,” in *6th Workshop on Detection and Classification of Acoustic Scenes and Events 2020 (DCASE)*, 2021, pp. 1–5.
- [33] K. Dohi, T. Nishida, H. Purohit, R. Tanabe, T. Endo, M. Yamamoto, Y. Nikaido, and Y. Kawaguchi, “MIMII DG: sound dataset for malfunctioning industrial machine investigation and inspection for domain generalization task,” in *7th Workshop on Detection and Classification of Acoustic Scenes and Events 2020 (DCASE)*. Tampere University, 2022, pp. 26–30.
- [34] N. Harada, D. Niizumi, D. Takeuchi, Y. Ohishi, and M. Yasuda, “ToyADMOS2+: New toyadmos data and benchmark results of the first-shot anomalous sound event detection baseline,” in *8th Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. Tampere University, 2023, pp. 41–45.
- [35] D. K. McClish, “Analyzing a portion of the ROC curve,” *Medical decision making*, vol. 9, no. 3, pp. 190–195, 1989.
- [36] K. Wilkinghoff, “Fraunhofer FKIE submission for task 2: First-shot unsupervised anomalous sound detection for machine condition monitoring,” DCASE2023 Challenge, Tech. Rep., 2023.
- [37] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations (ICLR)*, Y. Bengio and Y. LeCun, Eds., 2015.
- [38] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. JMLR Proceedings, vol. 9. JMLR.org, 2010, pp. 249–256.
- [39] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “Mixup: Beyond empirical risk minimization,” in *6th International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2018.
- [40] W. Junjie, W. Jiajun, C. Shengbing, S. Yong, and L. Mengyuan, “Anomalous sound detection based on self-supervised learning,” DCASE2023 Challenge, Tech. Rep., 2023.
- [41] Z. Lv, B. Han, Z. Chen, Y. Qian, J. Ding, and J. Liu, “Unsupervised anomalous detection based on unsupervised pretrained models,” DCASE2023 Challenge, Tech. Rep., 2023.
- [42] A. Jiang, Q. Hou, J. Liu, P. Fan, J. Ma, C. Lu, Y. Zhai, Y. Deng, and W.-Q. Zhang, “Thuee system for first-shot unsupervised anomalous sound detection for machine condition monitoring,” DCASE2023 Challenge, Tech. Rep., 2023.
- [43] J. Yafei, B. Jisheng, and H. Siwei, “Unsupervised abnormal sound detection based on machine condition mixup,” DCASE2023 Challenge, Tech. Rep., 2023.
- [44] Y. Zhou and Y. Long, “Attribute classifier with imbalance compensation for anomalous sound detection,” DCASE2023 Challenge, Tech. Rep., 2023.
- [45] J. Tian, H. Zhang, Q. Zhu, F. Xiao, H. Liu, X. Mei, Y. Liu, W. Wang, and J. Guan, “First-shot anomalous sound detection with gmm clustering and finetuned attribute classification using audio pretrained model,” DCASE2023 Challenge, Tech. Rep., 2023.
- [46] L. Wang, F. Chu, Y. Zhou, S. Wang, Z. Yan, S. Xu, Q. Wu, M. Cai, J. Pan, Q. Wang, J. Du, T. Gao, X. Fang, and L. Zou, “First-shot unsupervised anomalous sound detection using attribute classification and conditional autoencoder,” DCASE2023 Challenge, Tech. Rep., 2023.
- [47] N. Harada, D. Niizumi, D. Takeuchi, Y. Ohishi, and M. Yasuda, “First-shot anomaly detection for machine condition monitoring: A domain generalization baseline,” in *31st European Signal Processing Conference EUSIPCO*. IEEE, 2023.
- [48] W. JiaJun, “Self-supervised representation learning for first-shot unsupervised anomalous sound detection,” DCASE2023 Challenge, Tech. Rep., June 2023.

A.1.7 *Key publication 7*

Kevin Wilkinghoff. “Self-Supervised Learning for Anomalous Sound Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 276–280. DOI: [10.1109/ICASSP48485.2024.10447156](https://doi.org/10.1109/ICASSP48485.2024.10447156).
© 2024 IEEE.

SELF-SUPERVISED LEARNING FOR ANOMALOUS SOUND DETECTION

Kevin Wilkinghoff

Fraunhofer FKIE, Fraunhoferstraße 20, 53343 Wachtberg, Germany
kevin.wilkinghoff@ieee.org

ABSTRACT

State-of-the-art anomalous sound detection (ASD) systems are often trained by using an auxiliary classification task to learn an embedding space. Doing so enables the system to learn embeddings that are robust to noise and are ignoring non-target sound events but requires manually annotated meta information to be used as class labels. However, the less difficult the classification task becomes, the less informative are the embeddings and the worse is the resulting ASD performance. A solution to this problem is to utilize self-supervised learning (SSL). In this work, feature exchange (FeatEx), a simple yet effective SSL approach for ASD, is proposed. In addition, FeatEx is compared to and combined with existing SSL approaches. As the main result, a new state-of-the-art performance for the DCASE2023 ASD dataset is obtained that outperforms all other published results on this dataset by a large margin.

Index Terms— self-supervised learning, anomalous sound detection, domain generalization, machine listening

1. INTRODUCTION

In contrast to supervised learning, self-supervised learning (SSL) [1] does not require manually annotated class labels. Instead, data is augmented in different, specifically chosen ways, each defining another artificially created class and a model is taught to discriminate between these classes. The underlying assumption is that the model needs to understand the structure of the data to correctly predict the artificially introduced classes. SSL is a type of unsupervised learning and has been successfully applied to learning speech representations [2] or general purpose audio representations [3, 4] using large datasets of unlabeled data.

SSL has also been applied to anomalous sound detection (ASD): In [5], combinations of pitch-shifting and time-stretching are used to create additional classes. [6] uses linear combinations of similar target sounds, created by applying mixup [7], as pseudo-anomalous classes. Statistics exchange (StatEx) [8] is an SSL approach that mixes first- and second-order statistics of time-frequency representations to create new classes. In [9], a modified version of variance-invariance-covariance regularization [10], called domain generalization mixup, is used to pre-train autoencoders. Note that some works on ASD use the term SSL for supervised learning of embeddings with auxiliary classification tasks [11, 12]. Since classification tasks require manually labeled data whereas SSL does not require any manual annotations, we will use two different terms.

In [13], it was shown that class labels alone are very beneficial to detect anomalous sounds in noisy conditions as the model learns to closely monitor the target sounds. When not using class labels, many non-target sounds contained in an acoustic scene have a much stronger impact than subtle changes of the target sounds that need to be detected in order to identify anomalous sounds. However, as

the available classes become less similar to each other, the embeddings acquired through the discrimination of these classes capture less meaningful information. In these cases, also utilizing SSL is a very promising approach to increase the degree of information being captured and thus is expected to improve the ASD performance.

The goal of this work is to investigate different approaches of using SSL for ASD. The contributions are the following: First, two existing SSL approaches for ASD, namely mixup and StatEx, are reviewed. Second, feature exchange (FeatEx), a novel SSL approach for ASD, and a combination with the other two SSL approaches are proposed. In experimental evaluations conducted on the DCASE2022 ASD dataset and the DCASE2023 ASD dataset, it is shown that the proposed approach improves the performance over a baseline system not using any SSL. As a result, a new state-of-the-art performance significantly outperforming all published ASD results is obtained on the DCASE2023 ASD dataset¹.

2. STATE-OF-THE-ART BASELINE SYSTEM

Throughout this work, the ASD system presented in our prior work [15], which is trained in a supervised manner by using an auxiliary classification task, is used as a baseline system. Our goal is to improve the performance obtained with this system by applying SSL approaches. This system ranked 4th in the DCASE2023 Challenge [14] and the winning team of the challenge [16] extended this system by adding an attention mechanism to the embedding model. This shows that the ASD system can be considered state-of-the-art, which justifies choosing it as a baseline system.

An overview of the baseline system can be found in Figure 1. The system is based on learning discriminative embeddings by using all available meta information, in this case all combinations of machine types, machine IDs and attribute information, as classes. To this end, two convolutional sub-networks are used and their outputs are concatenated to obtain a single embedding. One sub-network utilizes the full magnitude frequency spectrum as an input representation to ensure the highest possible frequency resolution. The other sub-network uses magnitude spectrograms while subtracting the temporal mean to remove static frequency information and make the input representation more different from the other one. The neural network is trained for 10 epochs using a batch size of 64 by minimizing the angular margin loss sub-cluster AdaCos [17] with 16 sub-clusters. To improve the resulting ASD performance, no bias terms are used in any layer of the networks and the cluster centers are randomly initialized and not adapted during training. Apart from mixup, no other data augmentation technique is used. As a back-end, k-means is applied to the embeddings obtained with the normal training samples of the source domain, for which many training sam-

¹An open-source implementation of the system is available at: <https://github.com/wilkinghoff/ssl4asd>

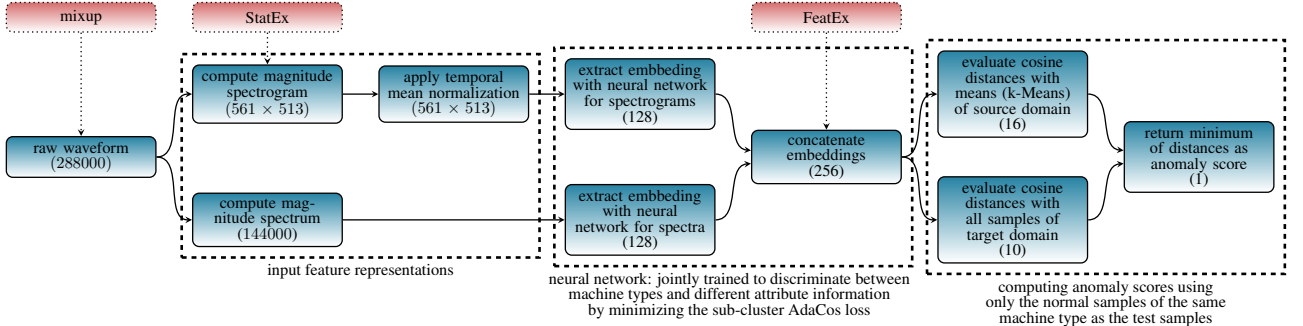


Fig. 1. Structure of the baseline system (blue boxes). SSL approaches are colored in red. Representation size in each step is given in brackets. This figure is adapted from [14] and originally adapted from [15].

ples are available. The smallest cosine distance to these means and all samples of the target domain, which differs from the target domain and for which only very few training samples are available, is used as an anomaly score. More details about this baseline system can be found in [15].

3. SELF-SUPERVISED LEARNING APPROACHES

In this section, two SSL approaches for training ASD systems are reviewed, namely mixup [7] and StatEx [8]. Furthermore, a third approach, called FeatEx, is proposed and described in detail. Last but not least, a combined SSL approach, which jointly uses all three discussed approaches, is presented.

3.1. Mixup

The data augmentation technique mixup [7], which uses linear interpolations between two training samples and their corresponding categorical labels, is widely applied for ASD [14, 16, 18–20]. When applying mixup, two randomly chosen training samples x_1, x_2 and their corresponding categorical class labels $y_1, y_2 \in [0, 1]^{N_{\text{classes}}}$ with $N_{\text{classes}} \in \mathbb{N}$ denoting the number of classes are combined by setting

$$\begin{aligned} x_{\text{new}} &= \lambda x_1 + (1 - \lambda)x_2 \\ y_{\text{new}} &= \lambda y_1 + (1 - \lambda)y_2 \end{aligned}$$

using a random mixing coefficient $\lambda \in [0, 1]$. Although mixup is just a data augmentation technique that does not introduce new classes, it can also be seen as a form of SSL that requires class labels because the supervised training objective is essentially extended to also predicting the mixing coefficient in addition to the original classes. In [8, 14], mixup was used to create additional pseudo-anomalous classes by treating mixed-up samples as belonging to other classes as non-mixed samples, similar to the approach proposed in [5]. In our experiments, this approach did not improve the ASD performance over applying mixup regularly but even degraded the performance for some machine types. It is also possible to only predict the mixing coefficient by ignoring the class labels, making mixup a purely self-supervised approach. However, for noisy audio data it is highly beneficial to utilize all available meta information for classification in order to teach the system to closely monitor the machine sounds of interest and ignore the background noise and other non-target events [13]. Throughout this work, we used a probability of 100% for applying mixup when training the baseline system, and a probability of 50% when also applying any other SSL approach.

3.2. Statistics exchange

The idea of StatEx [8] is to artificially create new classes of pseudo anomalies by exchanging the first- and second-order statistics of the time-frequency representations of two training samples x_1, x_2 along the temporal or frequency dimension. Mathematically, this corresponds to generating a new sample $x_{\text{new}} \in \mathbb{R}^{T \times F}$ by setting

$$x_{\text{new}} = \frac{x_1 - \mu_1}{\sigma_1} \sigma_2 + \mu_2$$

where $x_1, x_2 \in \mathbb{R}^{T \times F}$ denote the time-frequency representations of two random training samples. μ_1, μ_2 denote the first-order statistics of these samples along the time or frequency dimension, and σ_1, σ_2 denote the second-order statistics along the same dimension. Each possible combination of classes defines a new class, increasing the original number of classes $N_{\text{classes}} \in \mathbb{N}$ by a quadratic term N_{classes}^2 .

In this work, we used a variant of StatEx with the following modifications: For the sake of simplicity, we always use the complete frequency band and all time steps to calculate the statistics whereas in the original definition subbands are used [8]. Third, we teach the model to predict the class of the original sample x_1 and the class of the other sample x_2 , whose statistics are used. For categorical class labels y_1, y_2 , we do this by concatenating the labels:

$$y_{\text{new}} = (\mathbf{0}, 0.5 \cdot y_1, 0.5 \cdot y_2) \in [0, 1]^{3N_{\text{classes}}}$$

where $\mathbf{0} = (0, \dots, 0) \in [0, 1]^{N_{\text{classes}}}$. Hence, the number of classes is only tripled and thus the number of parameters, which, due to the cluster centers, proportionally increases with the number of classes, does not explode. Furthermore, this enables a simple combination with other data augmentation techniques, for which more than a single class are assigned to each sample, such as mixup. In [15], it has been shown that removing the temporal mean from the spectrograms improves the resulting ASD performance. Thus, in the variant used in this paper, we applied StatEx to the frequency axis, i.e. we only used temporal StatEx. Furthermore, two feature branches are used in the baseline system. Hence, only temporal StatEx has been applied to the spectrogram representations. Throughout this work, we used a probability of 50% for applying StatEx during training and also applied mixup. In case StatEx is not applied, the new label of training sample $x_{\text{new}} = x_1$ is set to

$$y_{\text{new}} = (y_1, \mathbf{0}, \mathbf{0}) \in [0, 1]^{3N_{\text{classes}}}.$$

In addition, we used trainable cluster centers for the newly introduced classes. These particular choices are also justified through ablation studies carried out in subsection 4.3.

Table 1. Harmonic means of AUCs and pAUCs taken over all machine IDs obtained when using different SSL approaches. Highest AUCs and pAUCs in each row are highlighted in bold letters. Arithmetic mean and standard deviation over five independent trials are shown.

dataset	split	domain	baseline [15]		StatEx [8] variant		FeatEx		regular and StatEx [8] variant		regular and FeatEx		proposed approach	
			AUC	pAUC	AUC	pAUC	AUC	pAUC	AUC	pAUC	AUC	pAUC	AUC	pAUC
DCASE2022	dev	source	84.2 ± 0.8%	76.5 ± 0.9%	80.5 ± 1.8%	69.5 ± 1.9%	82.1 ± 0.9%	72.8 ± 0.8%	85.2 ± 0.9%	77.5 ± 1.2%	85.1 ± 0.9%	76.3 ± 1.8%	86.0 ± 0.9%	77.6 ± 0.8%
DCASE2022	dev	target	78.5 ± 0.9%	62.5 ± 0.9%	75.3 ± 1.7%	60.0 ± 0.9%	77.2 ± 1.0%	62.8 ± 0.6%	78.9 ± 0.9%	63.2 ± 1.6%	77.9 ± 0.9%	62.7 ± 0.8%	78.2 ± 0.7%	64.4 ± 1.1%
DCASE2022	dev	mixed	81.4 ± 0.7%	66.6 ± 0.9%	76.4 ± 1.5%	62.4 ± 1.2%	78.5 ± 0.6%	65.2 ± 0.6%	82.2 ± 0.6%	67.0 ± 1.0%	81.6 ± 0.7%	67.0 ± 0.9%	82.5 ± 0.8%	68.2 ± 1.1%
DCASE2022	eval	source	76.8 ± 0.8%	65.8 ± 0.2%	74.2 ± 0.6%	61.6 ± 1.4%	76.3 ± 0.9%	64.5 ± 1.2%	76.9 ± 0.4%	65.8 ± 0.9%	78.1 ± 0.4%	67.0 ± 1.1%	77.7 ± 0.8%	67.0 ± 0.5%
DCASE2022	eval	target	69.8 ± 0.5%	59.7 ± 1.1%	70.6 ± 0.5%	59.0 ± 0.7%	72.3 ± 0.6%	61.0 ± 0.7%	71.2 ± 0.3%	60.3 ± 0.7%	72.2 ± 0.4%	61.3 ± 0.5%	71.6 ± 1.0%	61.2 ± 0.9%
DCASE2022	eval	mixed	73.4 ± 0.5%	59.8 ± 0.8%	72.2 ± 0.3%	58.2 ± 0.7%	73.9 ± 0.5%	60.0 ± 0.9%	73.9 ± 0.3%	59.9 ± 0.6%	74.9 ± 0.4%	61.5 ± 0.6%	74.2 ± 0.3%	61.2 ± 0.3%
DCASE2023	dev	source	69.8 ± 1.8%	60.9 ± 0.9%	67.8 ± 1.5%	59.2 ± 0.8%	68.4 ± 1.0%	60.2 ± 0.5%	70.3 ± 1.8%	62.0 ± 1.6%	72.9 ± 2.0%	63.0 ± 1.3%	71.2 ± 1.6%	62.7 ± 1.3%
DCASE2023	dev	target	72.3 ± 1.8%	55.6 ± 0.9%	69.7 ± 1.7%	54.7 ± 1.1%	74.4 ± 0.7%	57.6 ± 1.0%	72.2 ± 1.4%	56.2 ± 1.2%	75.7 ± 0.8%	57.0 ± 1.6%	75.0 ± 1.5%	56.1 ± 1.4%
DCASE2023	dev	mixed	71.3 ± 0.6%	56.1 ± 0.8%	69.0 ± 1.2%	55.7 ± 1.0%	71.7 ± 0.4%	57.5 ± 0.7%	71.2 ± 0.7%	57.0 ± 1.4%	74.4 ± 1.0%	58.0 ± 1.4%	73.1 ± 0.9%	57.3 ± 0.6%
DCASE2023	eval	source	72.5 ± 0.8%	62.4 ± 1.2%	70.0 ± 1.2%	59.7 ± 0.9%	69.3 ± 2.0%	59.3 ± 1.2%	72.4 ± 2.4%	62.4 ± 1.3%	75.9 ± 1.0%	62.9 ± 1.3%	75.5 ± 0.8%	64.5 ± 0.6%
DCASE2023	eval	target	63.1 ± 2.6%	57.5 ± 0.8%	66.7 ± 1.8%	58.4 ± 0.7%	69.1 ± 1.3%	59.0 ± 1.3%	65.9 ± 1.9%	59.0 ± 1.4%	66.5 ± 1.9%	58.3 ± 1.0%	68.7 ± 2.2%	59.3 ± 0.7%
DCASE2023	eval	mixed	67.9 ± 1.0%	58.8 ± 0.8%	65.8 ± 0.6%	57.1 ± 0.8%	68.1 ± 1.2%	58.1 ± 0.9%	69.5 ± 1.8%	60.7 ± 0.9%	71.1 ± 1.1%	60.1 ± 1.3%	72.6 ± 0.7%	61.6 ± 0.5%

3.3. Feature exchange

Look, listen and learn (L3) embeddings [21–23] are trained by using an audio and a video subnetwork, and predicting whether a video frame and an audio segment with a length of one second belong together or not. In [24], it was shown empirically that using these pre-trained embeddings does not lead to a better ASD performance than directly training an embedding model. When comparing multiple pre-trained embeddings, self-supervised embeddings such as L3-embeddings appear to outperform supervised embeddings. This motivates to develop a similar SSL approach for learning embeddings using only audio data.

As the baseline system also consists of two sub-networks, both utilizing different input feature representations, a similar SSL approach can be used, which we will call feature exchange (FeatEx). Let $e_1 = (e_1^1, e_1^2) \in \mathbb{R}^{2D}$, $e_2 = (e_2^1, e_2^2) \in \mathbb{R}^{2D}$ with $D = 128$ denote the concatenated embeddings of both sub-networks belonging to two random training samples x_1 and x_2 , and let y_1, y_2 denote their corresponding categorical class labels. Then, define a new embedding and its label by setting

$$e_{\text{new}} = (e_1^1, e_2^2) \in \mathbb{R}^{2D}$$

$$y_{\text{new}} = (\mathbf{0}, 0.5 \cdot y_1, 0.5 \cdot y_2) \in [0, 1]^{3N_{\text{classes}}}$$

where $N_{\text{classes}} \in \mathbb{N}$ denotes the number of the original classes and $\mathbf{0} = (0, \dots, 0) \in [0, 1]^{N_{\text{classes}}}$. Hence, as for the StatEx variant, the number of classes is tripled. When applying FeatEx, the network also needs to learn whether the embeddings of the sub-networks belong together or not resulting in more information being captured. Again, a combination with mixup, a probability of 50% for applying FeatEx during training and trainable cluster centers for the newly introduced classes have been used throughout this work.

3.4. Combining supervised and self-supervised losses

In [15], it was shown that not adapting randomly initialized cluster centers during training improves the resulting ASD performance. Experimentally, we found that the SSL approaches performed better with trainable cluster centers. To ensure that only the cluster centers of the SSL loss belonging to the original classes are non-trainable, we also used the regular supervised loss of the baseline system with non-trainable cluster centers as an equally weighted loss. Since the classes introduced by the SSL approaches are dividing the original classes into sub-classes, this can also be seen as a form of disentangled learning [25]. Furthermore, we propose to use a combination of mixup with the regular loss as well as StatEx and FeatEx in a single loss function since all SSL approaches are different. Hence, the total

loss $\mathcal{L}_{\text{total}}(x, y)$ of a sample x with categorical label y is given by

$$\mathcal{L}_{\text{total}}(x, y) = \mathcal{L}(x, y) + \mathcal{L}(x_{\text{new}}, y_{\text{new}})$$

with x_{new} and y_{new} being defined by sequentially applying all the SSL approaches as described in the previous sections and \mathcal{L} denoting the categorical crossentropy. As a result, the number of classes is increased multiplicatively by a factor of $3 \cdot 3 = 9$. In the following, this is called the *proposed approach*.

4. EXPERIMENTAL RESULTS

4.1. Datasets

For all experiments conducted in this work, the DCASE2022 [26] and the DCASE2023 ASD dataset [27] are used. Both datasets are designed for semi-supervised ASD in machine condition monitoring and contain noisy recordings of machine sounds of various types taken from ToyAdmos2 [28] and MIMII-DG [29]. For training, only normal sounds and additional meta information such as the machine types and parameter settings of the machines, called attribute information, are available. Furthermore, both datasets are designed for domain generalization and thus consist of data from a source domain with 1000 training samples for each machine id and a target domain with only 10 training samples that somehow differs by changing a parameter setting of the target machine or the background noise. The task is to discriminate between normal and anomalous samples regardless of the domain a sample belongs to.

Both datasets are divided into a development and an evaluation set, each consisting of a training subset with only normal data samples and a test subset containing normal and anomalous samples. The DCASE2022 ASD dataset consists of recordings belonging to seven different machine types, each with three different machine IDs contained in the development set and another three machine IDs contained in the evaluation set. For the DCASE2023 ASD dataset, there are 14 different machine types. The machine types belonging to the development and evaluation set are mutually exclusive, and there is only one machine ID for each machine type. Hence, the classification task is much easier than for the DCASE2022 dataset and thus learning informative embeddings by solving an auxiliary classification task is much more difficult for the DCASE2023 dataset, which motivates to also utilize SSL for training the ASD system.

4.2. Comparison of SSL approaches

As a first experiment, different SSL approaches are compared to the baseline performance obtained by not using additional SSL losses. The results can be found in Table 1 and the following observations

Table 2. Harmonic means of AUCs and pAUCs taken over all machine types obtained on the DCASE2023 dataset by modifying design choices of the proposed approach. Arithmetic mean and standard deviation over five independent trials are shown.

split	domain	SSL loss without class labels		non-trainable class centers		no TMN and full StatEx	
		AUC	pAUC	AUC	pAUC	AUC	pAUC
dev	source	70.8 ± 1.5%	63.2 ± 1.1%	71.5 ± 0.9%	64.8 ± 1.9%	70.9 ± 0.7%	61.0 ± 1.5%
	target	74.7 ± 1.5%	58.1 ± 1.6%	74.0 ± 2.0%	56.7 ± 1.0%	72.1 ± 1.4%	55.2 ± 1.0%
	mixed	72.3 ± 1.2%	57.9 ± 1.3%	71.6 ± 1.1%	57.7 ± 0.7%	71.3 ± 0.7%	55.6 ± 0.9%
eval	source	73.5 ± 2.4%	63.8 ± 0.6%	74.2 ± 0.7%	63.9 ± 1.3%	73.8 ± 1.3%	62.4 ± 1.5%
	target	62.1 ± 1.5%	57.7 ± 0.9%	58.2 ± 3.3%	57.3 ± 0.9%	66.9 ± 2.4%	58.5 ± 1.9%
	mixed	68.6 ± 1.2%	59.1 ± 0.7%	65.0 ± 0.9%	57.7 ± 0.6%	70.9 ± 0.8%	59.9 ± 0.8%

can be made: First, it can be seen that the proposed FeatEx loss performs significantly better than the StatEx loss on both datasets. Second, only using a loss based on StatEx leads to slightly worse performance than the performance obtained with the baseline system while only using the FeatEx loss lead to slightly better performance than the baseline system for most dataset splits. However, combining the regular loss with one of the SSL losses improves performance over both individual losses, especially on the DCASE2023 dataset. As stated before, the most likely reason is that, compared to the DCASE2022 dataset, there is only one specific machine for each machine type and thus the classification task is much easier resulting in less informative embeddings that are less sensitive to anomalies. Hence, SSL is required as a form of regularization to learn non-trivial mappings for each class resulting in more informative embeddings that enable the system to detect subtle deviations from normal data. As a last observation, it can be seen that combining all SSL approaches into a single loss, slightly improves the resulting performance for some dataset splits while decreasing the performance for other dataset splits. Overall, the positive effects seem to be slightly greater than the negative ones but are only marginal.

4.3. Ablation studies

To show that the design choices of the proposed approach actually optimize the ASD performance, three ablation studies have been conducted on the DCASE2023 dataset. More concretely, it was verified whether 1) not using the class labels for the SSL losses, 2) using non-trainable class centers for the SSL losses or 3) not using temporal mean normalization (TMN) but also applying StatEx to the temporal axis improves the performance. When comparing the results, as shown in Table 2, to the original ones contained in Table 1, it can be seen that altering the proposed approach in any of the three ways degrades ASD performance, especially on the evaluation set. This adds confidence to the design of the proposed SSL approach.

4.4. Comparison to other published systems

As a last experiment, the proposed system was compared to the ten top-performing systems submitted to the DCASE2023 Challenge. To have a fair comparison, we used an ensemble obtained by re-training the system five times and taking the mean of all anomaly scores. The results can be found in Figure 2. It can be seen that our proposed system outperforms all other published systems by a significant margin and thus reaches a new state-of-the-art performance. Note that the system ranked fourth [14] in the DCASE2023 Challenge is actually the same system as the baseline system of this work and the system ranked first [16] is a modified version of this baseline system.

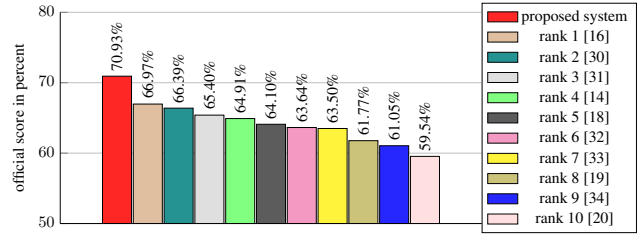


Fig. 2. Comparison between presented and ten top-performing systems of the DCASE Challenge 2023.

5. CONCLUSION

In this work, applying SSL to ASD was investigated. To this end, mixup and StatEx were reviewed, and a novel SSL approach for ASD, called FeatEx was proposed. All three approaches were combined into a single loss function for training an outlier exposed ASD system. In experiments conducted on the DCASE2022 and DCASE2023 ASD datasets, it was shown that FeatEx outperforms the existing SSL approaches, and that applying SSL to ASD is highly beneficial. As a result, a new state-of-the-art performance on the DCASE2023 ASD dataset was obtained outperforming all other published systems by a large margin.

6. ACKNOWLEDGMENTS

The author would like to thank Fabian Fritz, Lukas Henneke and Frank Kurth for their valuable feedback.

7. REFERENCES

- [1] Shuo Liu, Adria Mallol-Ragolta, Emilia Parada-Cabaleiro, Kun Qian, Xin Jing, Alexander Kathan, Bin Hu, and Björn W. Schuller, “Audio self-supervised learning: A survey,” *Patterns*, vol. 3, no. 12, pp. 100616, 2022.
- [2] Abdelrahman Mohamed, Hung-yi Lee, Lasse Borgholt, Jakob D. Havtorn, Joakim Edin, Christian Igel, Katrin Kirchhoff, Shang-Wen Li, Karen Livescu, Lars Maaløe, Tara N. Sainath, and Shinji Watanabe, “Self-supervised speech representation learning: A review,” *IEEE J. Sel. Top. Signal Process.*, vol. 16, no. 6, pp. 1179–1210, 2022.
- [3] Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino, “BYOL for audio: Self-supervised learning for general-purpose audio representation,” in *IJCNN*. 2021, pp. 1–8, IEEE.
- [4] Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino, “BYOL for audio: Exploring pre-trained general-purpose audio representations,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 31, pp. 137–151, 2023.
- [5] Tadanobu Inoue, Phongtharin Vinayavekhin, Shu Morikuni, Shiqiang Wang, Tuan Hoang Trong, David Wood, Michiaki Tatsubori, and Ryuki Tachibana, “Detection of anomalous sounds for machine condition monitoring using classification confidence,” in *DCASE*, 2020, pp. 66–70.
- [6] Jose A. Lopez, Hong Lu, Paulo Lopez-Meyer, Lama Nachman, Georg Stemmer, and Jonathan Huang, “A speaker recognition approach to anomaly detection,” in *DCASE*, 2020, pp. 96–99.

- [7] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz, “Mixup: Beyond empirical risk minimization,” in *ICLR*, 2018.
- [8] Han Chen, Yan Song, Zhu Zhuo, Yu Zhou, Yu-Hong Li, Hui Xue, and Ian McLoughlin, “An effective anomalous sound detection method based on representation learning with simulated anomalies,” in *ICASSP*. IEEE, 2023.
- [9] Ismail Nejjar, Jean Meunier-Pion, Gaëtan Frusque, and Olga Fink, “DG-Mix: Domain generalization for anomalous sound detection based on self-supervised learning,” in *DCASE*. 2022, Tampere University.
- [10] Adrien Bardes, Jean Ponce, and Yann LeCun, “VICReg: Variance-invariance-covariance regularization for self-supervised learning,” in *ICLR*. 2022, OpenReview.net.
- [11] Ritwik Giri, Srikanth V. Tenneti, Fangzhou Cheng, Karim Helwani, Umut Isik, and Arvinth Krishnaswamy, “Self-supervised classification for detecting anomalous sounds,” in *DCASE*, 2020, pp. 46–50.
- [12] Kota Dohi, Takashi Endo, Harsh Purohit, Ryo Tanabe, and Yohei Kawaguchi, “Flow-based self-supervised density estimation for anomalous sound detection,” in *ICASSP*. 2021, pp. 336–340, IEEE.
- [13] Kevin Wilkinghoff and Frank Kurth, “Why do angular margin losses work well for semi-supervised anomalous sound detection?,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 32, pp. 608–622, 2024.
- [14] Kevin Wilkinghoff, “Fraunhofer FKIE submission for task 2: First-shot unsupervised anomalous sound detection for machine condition monitoring,” Tech. Rep., DCASE2023 Challenge, 2023.
- [15] Kevin Wilkinghoff, “Design choices for learning embeddings from auxiliary tasks for domain generalization in anomalous sound detection,” in *ICASSP*. 2023, IEEE.
- [16] Wang Junjie, Wang Jiajun, Chen Shengbing, Sun Yong, and Liu Mengyuan, “Anomalous sound detection based on self-supervised learning,” Tech. Rep., DCASE2023 Challenge, 2023.
- [17] Kevin Wilkinghoff, “Sub-cluster AdaCos: Learning representations for anomalous sound detection,” in *IJCNN*. 2021, IEEE.
- [18] Jia Yafei, Bai Jisheng, and Huang Siwei, “Unsupervised abnormal sound detection based on machine condition mixup,” Tech. Rep., DCASE2023 Challenge, 2023.
- [19] Lei Wang, Fan Chu, Yuxuan Zhou, Shuxian Wang, Zulong Yan, Shifan Xu, Qing Wu, Mingqi Cai, Jia Pan, Qing Wang, Jun Du, Tian Gao, Xin Fang, and Liang Zou, “First-shot unsupervised anomalous sound detection using attribute classification and conditional autoencoder,” Tech. Rep., DCASE2023 Challenge, 2023.
- [20] Wang JiaJun, “Self-supervised representation learning for first-shot unsupervised anomalous sound detection,” Tech. Rep., DCASE2023 Challenge, June 2023.
- [21] Relja Arandjelovic and Andrew Zisserman, “Look, listen and learn,” in *ICCV*. 2017, pp. 609–617, IEEE Computer Society.
- [22] Relja Arandjelovic and Andrew Zisserman, “Objects that sound,” in *ECCV*. 2018, vol. 11205 of *Lecture Notes in Computer Science*, pp. 451–466, Springer.
- [23] Aurora Cramer, Ho-Hsiang Wu, Justin Salamon, and Juan Pablo Bello, “Look, listen, and learn more: Design choices for deep audio embeddings,” in *ICASSP*. 2019, pp. 3852–3856, IEEE.
- [24] Kevin Wilkinghoff and Fabian Fritz, “On using pre-trained embeddings for detecting anomalous sounds with limited training data,” in *EUSIPCO*. 2023, pp. 186–190, IEEE.
- [25] Satvik Venkatesh, Gordon Wichern, Aswin Shanmugam Subramanian, and Jonathan Le Roux, “Improved domain generalization via disentangled multi-task learning in unsupervised anomalous sound detection,” in *DCASE*. 2022, Tampere University.
- [26] Kota Dohi, Keisuke Imoto, Noboru Harada, Daisuke Niizumi, Yuma Koizumi, Tomoya Nishida, Harsh Purohit, Ryo Tanabe, Takashi Endo, Masaaki Yamamoto, and Yohei Kawaguchi, “Description and discussion on DCASE 2022 challenge task 2: Unsupervised anomalous sound detection for machine condition monitoring applying domain generalization techniques,” in *DCASE*. 2022, Tampere University.
- [27] Kota Dohi, Keisuke Imoto, Noboru Harada, Daisuke Niizumi, Yuma Koizumi, Tomoya Nishida, Harsh Purohit, Ryo Tanabe, Takashi Endo, and Yohei Kawaguchi, “Description and discussion on DCASE 2023 challenge task 2: First-shot unsupervised anomalous sound detection for machine condition monitoring,” in *DCASE*. 2023, pp. 31–35, Tampere University.
- [28] Noboru Harada, Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Masahiro Yasuda, and Shoichiro Saito, “Toy-ADMOS2: Another dataset of miniature-machine operating sounds for anomalous sound detection under domain shift conditions,” in *DCASE*, 2021, pp. 1–5.
- [29] Kota Dohi, Tomoya Nishida, Harsh Purohit, Ryo Tanabe, Takashi Endo, Masaaki Yamamoto, Yuki Nikaido, and Yohei Kawaguchi, “MIMII DG: sound dataset for malfunctioning industrial machine investigation and inspection for domain generalization task,” in *DCASE*. 2022, pp. 26–30, Tampere University.
- [30] Zhiqiang Lv, Bing Han, Zhengyang Chen, Yanmin Qian, Jiawei Ding, and Jia Liu, “Unsupervised anomalous detection based on unsupervised pretrained models,” Tech. Rep., DCASE2023 Challenge, 2023.
- [31] Anbai Jiang, Qijun Hou, Jia Liu, Pingyi Fan, Jitao Ma, Cheng Lu, Yuanzhi Zhai, Yufeng Deng, and Wei-Qiang Zhang, “Thuee system for first-shot unsupervised anomalous sound detection for machine condition monitoring,” Tech. Rep., DCASE2023 Challenge, 2023.
- [32] Yifan Zhou and Yanhua Long, “Attribute classifier with imbalance compensation for anomalous sound detection,” Tech. Rep., DCASE2023 Challenge, 2023.
- [33] Jiantong Tian, Hejing Zhang, Qiaoxi Zhu, Feiyang Xiao, Haohe Liu, Xinhao Mei, Youde Liu, Wenwu Wang, and Jian Guan, “First-shot anomalous sound detection with gmm clustering and finetuned attribute classification using audio pretrained model,” Tech. Rep., DCASE2023 Challenge, 2023.
- [34] Noboru Harada, Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, and Masahiro Yasuda, “First-shot anomaly detection for machine condition monitoring: A domain generalization baseline,” in *EUSIPCO*. 2023, pp. 191–195, IEEE.

A.1.8 *Key publication 8*

Kevin Wilkinghoff and Alessia Cornaggia-Urrigshardt. “TACos: Learning Temporally Structured Embeddings for Few-Shot Keyword Spotting with Dynamic Time Warping.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 9941–9945. DOI: [10.1109/ICASSP48485.2024.10445814](https://doi.org/10.1109/ICASSP48485.2024.10445814).

© 2024 IEEE.

The co-author of this publication contributed in the following ways: *Alessia Cornaggia-Urrigshardt* assisted with creating the dataset. She also applied [DTW](#), tested the [HFCC](#)-based approach and evaluated global and individual decision thresholds.

TACOS: LEARNING TEMPORALLY STRUCTURED EMBEDDINGS FOR FEW-SHOT KEYWORD SPOTTING WITH DYNAMIC TIME WARPING

Kevin Wilkinghoff¹ and Alessia Cornaggia-Urrigshardt¹

¹Fraunhofer FKIE, Fraunhoferstraße 20, 53343 Wachtberg, Germany
kevin.wilkinghoff@ieee.org, alessia.cornaggia-urrigshardt@fkie.fraunhofer.de

ABSTRACT

To segment a signal into blocks to be analyzed, few-shot keyword spotting (KWS) systems often utilize a sliding window of fixed size. Because of the varying lengths of different keywords or their spoken instances, choosing the right window size is a problem: A window should be long enough to contain all necessary information needed to recognize a keyword but a longer window may contain irrelevant information such as multiple words or noise and thus makes it difficult to reliably detect on- and offsets of keywords. We propose TACos, a novel angular margin loss for deriving two-dimensional embeddings that retain temporal properties of the underlying speech signal. In experiments conducted on KWS-DailyTalk, a few-shot KWS dataset presented in this work, using these embeddings as templates for dynamic time warping is shown to outperform using other representations or a sliding window and that using time-reversed segments of the keywords during training improves the performance.

Index Terms— keyword spotting, representation learning, angular margin loss, few-shot learning

1. INTRODUCTION

Keyword spotting (KWS) [1] is the task of detecting spoken instances of a few pre-defined keywords in audio recordings of possibly long duration. All other audio content should be ignored and thus KWS is inherently an open-set classification task. Additionally, for many KWS applications only very few training samples are available (few-shot classification [2]) and it is important to detect on- and offsets of detected keywords for further (manual) analysis of the content. Typical KWS applications are activating voice assistants [3,4], searching for content in large databases [5] or monitoring audio streams such as (radio) communication transmissions [6].

Many state-of-the-art KWS systems rely on segmenting audio signals and applying a neural network to extract discriminative embeddings for each segment that can be used to detect keywords [7,8]. For few-shot KWS, neural networks with a prototypical loss [9] are often used to learn an embedding function [10–12]. Similar approaches are used for few-shot detection of bioacoustic events [13] or sound events in general [14, 15]. Usually, a sliding window is applied to segment the signal into blocks of fixed size, in which keywords are searched. The chosen window size needs to be long enough to capture sufficient information for identifying a keyword. However, longer windows are likely to contain multiple keywords or, in case of a short keyword, too much irrelevant information thus degrading the performance. Additionally, precisely estimating on- and offsets of detected keywords is much more difficult with longer windows. Furthermore, the length of different keywords can strongly vary making it difficult to determine a suitable, fixed, window length.

In automatic speech recognition (ASR) [16], this problem is solved by using sequence-to-sequence losses such as the connectionist temporal classification loss [17]. However, training with such losses requires sufficient amounts of data making them unsuitable for a few-shot classification task. Although it is also possible to use a pre-trained ASR system [18, 19] or pre-trained ASR embeddings [20], this requires collecting many hours of training data recorded in similar acoustic environments and creates a large computational overhead. In [21], it has been proposed to save computational power by providing the network the ability to spike once a keyword is detected and immediately stop analyzing the remaining part of the input sequence. Classically, hand-crafted speech features such as human factor cepstral coefficients (HFCCs) [22] are used as two-dimensional templates for dynamic time warping (DTW). However, the performance of these unsupervised approaches quickly degrades for very short words or in difficult acoustic conditions. It is also possible to combine multiple KWS approaches: In [23], DTW and hand-crafted speech features are used to obtain training data for a neural network-based KWS system. In our prior work [24], two-dimensional embeddings, to be used as features for DTW, are trained using a neural network applied to windowed segments of audio signals. Still, the obtained embeddings are mostly constant over time and thus many problems resulting from using a sliding window persist.

The contributions of this work are the following: First and foremost, TACos, a novel loss function for learning embeddings that also capture the temporal structure, and a DTW-based few-shot KWS system are proposed. Second, the few-shot KWS dataset KWS-DailyTalk¹ based on the ASR dataset DailyTalk [25] is presented. In contrast to existing KWS datasets such as SpeechCommands [26], KWS-DailyTalk is an open-set classification dataset with isolated keywords as training data and complete spoken sentences as validation and test data. The proposed KWS system is shown to outperform systems using hand-crafted speech features, a sliding window or other KWS embeddings. Furthermore, it is shown that teaching the model to distinguish between regular and temporally reversed segments improves the performance.

2. METHODOLOGY

This work is based on prior work on learning two-dimensional embeddings for KWS with DTW [24]. A KWS system using these embeddings can be divided into a frontend for pre-processing the data, a neural network for extracting embeddings, and a backend for computing scores and finding keywords using DTW. To improve the quality of the embeddings, we propose 1) TACos, a novel angular margin loss that also considers the position of a segment in

¹<https://github.com/wilkinghoff/kws-dailytalk>

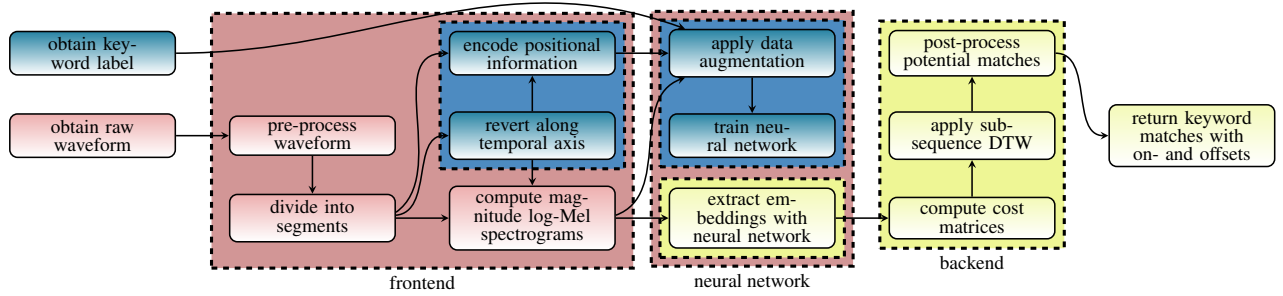


Fig. 1. Structure of the proposed KWS system. Blocks colored in blue are only used for training the system, blocks colored in yellow are only used for inference and blocks colored in red are used for training and inference.

a keyword, and 2) recognizing temporally reversed segments during training. In the following, each component of the KWS system will be reviewed and afterwards both proposed improvements will be presented. An overview of the resulting KWS system is depicted in Figure 1.

2.1. Review of embeddings for KWS with DTW

Frontend: First, all waveforms are converted to single-channel, normalized to an amplitude of 1, resampled to 16 kHz and high-pass filtered at 50 Hz. For the training samples containing isolated keywords, the waveforms are divided into overlapping segments with a length of $L_{\text{seg}} = 0.25$ s and an overlap of $\frac{L_{\text{seg}}}{5}$. During inference, a segment overlap of $\frac{256}{16000 \text{ Hz}}$ is used to increase the temporal resolution of the resulting embeddings. Furthermore, samples are padded with $\lfloor \frac{L_{\text{seg}} \cdot 16000}{2} \rfloor$ zeros on both sides to ensure that the centers of the extracted segments align with their temporal position in the audio signal. Segments shorter than L_{seg} are padded with zeros. From these, log-Mel magnitude spectrograms with 64 Mel bins are extracted using an STFT with Hanning-weighted windows of size 1024 and a hop size of 256.

Neural network: To extract two-dimensional embeddings, the modified ResNet architecture from [24] is used. This model consists of four times two residual blocks [27], each using convolutional layers with filters of size 3×3 , max-pooling for the frequency dimension and dropout with a probability of 20%, followed by a global max-pooling operation over the frequency dimension and a dense layer without activation function. Throughout the network, the same time dimension is kept by padding appropriately and not applying temporal pooling operations. The model, without the loss function, has only 713, 486 trainable parameters. As a loss function, the AdaCos loss [28], which is an angular margin loss for classification with a dynamically adaptive scale parameter, is used to discriminate between different keywords. Additional details can be found in [24].

The network is trained for 1000 epochs with a batch size of 32 using Adam [29]. Most keywords have different lengths resulting in a class imbalance due to a different total number of segments for each keyword class. To handle this, random oversampling is applied. For data augmentation, Mixup [30] with a mixing coefficient drawn from a uniform distribution and SpecAugment [31] are used. During training, random segments of the background noise recordings from SpeechCommands [26] are used as an additional “no speech” class.

Backend: The backend consists of applying sub-sequence DTW [32]. All embeddings belonging to different segments of the

same audio signal are combined by taking the mean of all individual frames of the time-frequency representation that overlap in time resulting in DTW templates. In a next step, cost matrices are computed by applying the pairwise cosine distance between the templates of test sentences and the templates extracted from individual training samples. We also experimented with computing Fréchet means of all templates belonging to the same keyword with the DTW barycenter averaging (DBA) algorithm [33] but this led to worse performance than using the templates of the individual training samples. To compute accumulated cost matrices, the DTW step sizes (2, 1), (1, 1) and (1, 2) are used. Note that computing the accumulated cost matrices can be parallelized by sweeping diagonally over the cost matrix. In total, the presented KWS approach is much faster than real-time. For each temporal position, a warping path is calculated and the corresponding accumulated cost is normalized with respect to the path length. The negative accumulated costs serve as matching scores that can be compared to a pre-defined threshold. Scores exceeding the threshold are considered as valid detections of a keyword and the start and end positions of the corresponding paths are returned as on- and offsets, respectively. If multiple detections overlap in time, all detections are shortened such that, at each position in time, only the single detection with the highest score is kept. Detections with less than half the duration of the training sample belonging to the detected keyword are discarded.

2.2. TACos loss function

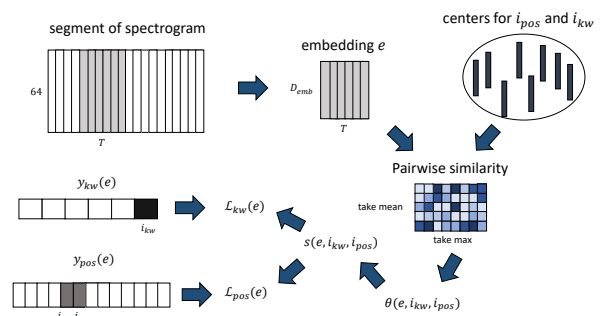


Fig. 2. Illustration of the TACos loss function.

The TACos loss function, illustrated in Figure 2, consists of a supervised loss \mathcal{L}_{kw} for predicting the keyword a given audio segment belongs to, which is the same loss as defined in [24], and a self-supervised loss \mathcal{L}_{pos} for predicting the relative position of this segment within a keyword. When only using a supervised loss, the resulting embeddings are mostly constant over time. The idea of introducing a positional loss is to force the network to learn two-dimensional embeddings that are changing over time and thus are more suitable to be used as templates for DTW.

Since the length of different keywords may vary substantially, the absolute position of a segment within a keyword has to be encoded relative to the length of the keyword to be able to use a fixed number of position classes for all keywords. Let $K \in \mathbb{N}$ be the total number of training samples, $N_{\text{seg}}(k) \in \mathbb{N}$ denote the number of segments belonging to the k th training sample and define $N_{\text{pos}} := \max_{k \in \{1, \dots, K\}} \{N_{\text{seg}}(k)\}$. Then, the relative positional encoding $y_{\text{pos}}(e_{k, i_{\text{seg}}}) \in [0, 1]^{N_{\text{pos}}}$ of embedding $e_{k, i_{\text{seg}}}$ belonging to segment $i_{\text{seg}} \in \{1, \dots, N_{\text{seg}}(k)\}$ of keyword sample k is obtained by setting

$$y_{\text{pos}}(e_{k, i_{\text{seg}}}, i_{\text{pos}}) = \frac{\mathbb{1}_{I_{\text{active}}(k, i_{\text{seg}})}(i_{\text{pos}})}{\sum_{j_{\text{pos}}=1}^{N_{\text{pos}}} \mathbb{1}_{I_{\text{active}}(k, i_{\text{seg}})}(j_{\text{pos}})}$$

with $I_{\text{active}}(k, i_{\text{seg}}) = [1 + \lceil \frac{(i_{\text{seg}}-1) \cdot N_{\text{pos}}}{N_{\text{seg}}(k)} \rceil, \lceil \frac{i_{\text{seg}} \cdot N_{\text{pos}}}{N_{\text{seg}}(k)} \rceil]$. Thus, for keyword samples shorter than the longest training sample, multiple positions may be set as active with equal probability resulting in soft labels for the position.

Let $N_{\text{kw}} \in \mathbb{N}$ denote the number of different keywords in the dataset and let $y_{\text{kw}}(e, i_{\text{kw}}) \in [0, 1]$ denote the categorical keyword labels of the samples. Let $T := \lceil L_{\text{seg}} \cdot \frac{16000}{256} \rceil \in \mathbb{N}$ be the time dimension and $D_{\text{emb}} \in \mathbb{N}$ be the embedding dimension of an embedding $e \in \mathbb{R}^{T \times D_{\text{emb}}}$ belonging to a single segment. Then, we define the cosine similarity of embedding e to keyword $i_{\text{kw}} \in \{1, \dots, N_{\text{kw}}\}$ and position $i_{\text{pos}} \in \{1, \dots, N_{\text{pos}}\}$ as a temporal mean given by

$$\theta(e, i_{\text{kw}}, i_{\text{pos}}) := \frac{1}{T} \sum_{t=1}^T \max_{i_{\text{cluster}}} \frac{\langle e(t), c(i_{\text{cluster}}, i_{\text{kw}}, i_{\text{pos}}) \rangle}{\|e(t)\|_2 \|c(i_{\text{cluster}}, i_{\text{kw}}, i_{\text{pos}})\|_2}$$

for trainable cluster centers $c \in \mathbb{R}^{N_{\text{cluster}} \times N_{\text{kw}} \times N_{\text{pos}} \times D_{\text{emb}}}$ with $N_{\text{cluster}} \in \mathbb{N}$, which do not have a temporal dimension. The corresponding softmax probability of embedding e belonging to keyword i_{kw} and position i_{pos} is defined as

$$s(e, i_{\text{kw}}, i_{\text{pos}}) := \frac{\exp(\hat{s} \cdot \theta(e, i_{\text{kw}}, i_{\text{pos}}))}{\sum_{j_{\text{kw}}=1}^{N_{\text{kw}}} \sum_{j_{\text{pos}}=1}^{N_{\text{pos}}} \exp(\hat{s} \cdot \theta(e, j_{\text{kw}}, j_{\text{pos}}))}$$

where $\hat{s} \in \mathbb{R}_+$ is the dynamically adaptive scale parameter as defined for the sub-cluster AdaCos loss in [34]. The probability of embedding e belonging to keyword i_{kw} is set to $\sum_{j_{\text{pos}}=1}^{N_{\text{pos}}} s(e, i_{\text{kw}}, j_{\text{pos}})$ and the probability of embedding e belonging to position i_{pos} is set to $\sum_{j_{\text{kw}}=1}^{N_{\text{kw}}} s(e, j_{\text{kw}}, i_{\text{pos}})$. Therefore, the loss functions for a single embedding e are equal to

$$\mathcal{L}_{\text{kw}}(e) := \sum_{i_{\text{kw}}=1}^{N_{\text{kw}}} y_{\text{kw}}(e, i_{\text{kw}}) \log \left(\sum_{i_{\text{pos}}=1}^{N_{\text{pos}}} s(e, i_{\text{kw}}, i_{\text{pos}}) \right)$$

$$\mathcal{L}_{\text{pos}}(e) := \sum_{i_{\text{pos}}=1}^{N_{\text{pos}}} y_{\text{pos}}(e, i_{\text{pos}}) \log \left(\sum_{i_{\text{kw}}=1}^{N_{\text{kw}}} s(e, i_{\text{kw}}, i_{\text{pos}}) \right)$$

and the TACos loss used for training the network is

$$\mathcal{L}_{\text{TAC}} := -\frac{1}{K} \sum_{k=1}^K \frac{1}{N_{\text{seg}}(k)} \sum_{i_{\text{seg}}=1}^{N_{\text{seg}}(k)} (\mathcal{L}_{\text{kw}}(e_{k, i_{\text{seg}}}) + \mathcal{L}_{\text{pos}}(e_{k, i_{\text{seg}}}))$$

Note that the cluster centers c significantly increase the total number of trainable parameters of the model. For all experiments in this work, $D_{\text{emb}} = 128$ and $N_{\text{cluster}} = 16$ were used.

2.3. Using temporally reversed segments

As a second modification, we propose to utilize temporally reversed versions of all keyword segments as additional training samples when training the embedding model. The idea is that the network has to be able to recognize the correct temporal order, i.e., we enforce that the reverse keyword is considered to be different from the non-reversed version. By doing so, the model has a harder task in correctly predicting the corresponding keyword and position of the segments. This leads to more informative embeddings and reduces the number of false detections. For each keyword class except for “no speech”, an additional unique label for the reversed keyword segments is introduced, almost doubling the number of different keyword classes. The position of the reversed segments and the segments not containing any speech are both encoded by using a uniform posterior probability over the position classes.

3. EXPERIMENTS

3.1. Dataset

For all experimental evaluations, KWS-DailyTalk based on the ASR dataset DailyTalk [25] was used. KWS-DailyTalk is a five-shot KWS dataset aimed at detecting 15 different keywords, namely “afternoon”, “airport”, “cash”, “credit card”, “deposit”, “dollar”, “evening”, “expensive”, “house”, “information”, “money”, “morning”, “night”, “visa” and “yuan”. The dataset consists of a training set with five isolated samples for each keyword and a duration of 39 seconds, as well as a validation and a test set, with an approximate duration of 10 minutes each, containing 156 and 157 sentences taken from dialogues, respectively. These sentences contain either none, a single, or multiple occurrences of the keywords as well as several other words that are not of interest but may cause false alarms. All keywords appear about 12 times each in the validation and the test set. The on- and offset of each keyword were manually annotated. Furthermore, it is ensured that a keyword sample used for training and the sentences of the validation and test set that also contain this keyword do not belong to the same conversation to make the KWS task more realistic and slightly more difficult. For all experimental evaluations, the event-based F-score (micro-averaged) as implemented in the *sed_eval* toolbox [35] was used. All hyperparameters of the KWS systems were tuned to maximize the performance on the validation set.

3.2. Baseline approaches

For comparison, the embeddings from [24] reviewed in subsection 2.1 and the following two baseline systems are used.

HFCCs: Instead of applying sub-sequence DTW to trained embeddings, HFCCs [22, 32] based on spectrograms with a window of 40 ms and a step size of 10 ms are used. These features were shown to outperform Mel-frequency cepstral coefficients in query-by-example KWS approaches. The DTW algorithm is the same as stated in subsection 2.1.

Table 1. Event-based, micro-averaged F-score, precision and recall obtained on KWS-DailyTalk with different KWS systems. Highest F-scores for each feature representation are highlighted with bold letters, overall highest F-scores are underlined.

KWS feature representation	reversed segments	threshold	obtained performance					
			validation set			test set		
			F-score	precision	recall	F-score	precision	recall
HFCCs	not applicable	global	60.52%	63.25%	58.01%	56.97%	61.54%	53.04%
HFCCs	not applicable	individual	64.71%	69.18%	60.77%	57.74%	62.58%	53.59%
embeddings (sliding)	not used	global	39.76%	43.71%	36.46%	38.35%	41.14%	35.91%
embeddings (sliding)	not used	individual	46.96%	49.39%	44.75%	40.44%	40.00%	40.88%
embeddings (sliding)	used	global	44.13%	62.00%	34.25%	44.83%	59.63%	35.91%
embeddings (sliding)	used	individual	55.43%	54.55%	56.35%	50.42%	51.14%	49.72%
embeddings (\mathcal{L}_{kw}) [24]	not used	global	56.04%	52.40%	60.22%	54.25%	53.80%	54.70%
embeddings (\mathcal{L}_{kw}) [24]	not used	individual	58.38%	57.14%	59.67%	53.04%	53.04%	53.04%
embeddings (\mathcal{L}_{kw})	used	global	64.58%	74.64%	56.91%	61.30%	69.72%	54.70%
embeddings (\mathcal{L}_{kw})	used	individual	66.12%	64.89%	67.40%	60.53%	57.79%	63.54%
embeddings ($\mathcal{L}_{kw} + \mathcal{L}_{pos}$)	not used	global	62.78%	75.78%	53.59%	64.65%	82.76%	53.04%
embeddings ($\mathcal{L}_{kw} + \mathcal{L}_{pos}$)	not used	individual	63.36%	63.19%	63.54%	63.31%	68.15%	59.12%
embeddings ($\mathcal{L}_{kw} + \mathcal{L}_{pos}$)	used	global	65.78%	82.50%	54.70%	70.47%	89.74%	58.01%
embeddings ($\mathcal{L}_{kw} + \mathcal{L}_{pos}$)	used	individual	69.44%	75.00%	64.64%	69.16%	79.29%	61.33%

Sliding window: As a third system, a sliding window based approach using a neural network trained with the standard AdaCos loss [28] is used. The network architecture is the same as presented in subsection 2.1 with the following modifications: For each residual block, the max-pooling operation is also applied to the temporal dimension and a flattening operation is applied before projecting onto the embedding space. To detect on- and offsets of keywords, the cosine distance of the resulting embeddings to each of the keyword-specific centers is calculated first. Then, these cosine distances are compared to a pre-defined threshold and a median filter with a size equal to the average number of segments over all training samples belonging to the corresponding keyword, rounded to the nearest odd natural number, is applied to the Boolean decision results. Start and end points of the resulting positive regions are adjusted by adding $-\frac{L_{seg}}{2}$ and $+\frac{L_{seg}}{2}$, respectively, and indicate on- and offsets of detected keywords.

3.3. Experimental results

The experimental results obtained on KWS-DailyTalk can be found in Table 1. First, it can be seen that using the proposed TACos loss leads to significantly better performance than when only using \mathcal{L}_{kw} , as done in [24], or using a sliding window based approach, particularly so on the test set. Moreover, both embeddings perform better than HFCCs despite having only 39 seconds of audio recordings available for training. A second major observation is that using temporally reversed segments for training the embedding model always improves the performance regardless of the chosen KWS approach. We want to emphasize that this can be observed although very powerful data augmentation techniques, namely Mixup and SpecAugment, are applied. Furthermore, tuning individual decision thresholds for each keyword class only improves the test performance when using a sliding window but not for the proposed approach for which the performance actually slightly decreases. Thus, another advantage is that one does not have to tune individual decision thresholds, which would be very impractical.

4. CONCLUSIONS

In this paper, TACos, a novel loss function for training neural networks to extract discriminative embeddings with temporal structure and a few-shot KWS-system based on DTW utilizing this loss were proposed. TACos consists of two components for learning the corresponding keyword of small speech segments as well as their relative position within a given keyword. To evaluate the performance of the KWS system, KWS-DailyTalk, an open-source dataset for few-shot keyword spotting based on DailyTalk, was presented. In experiments conducted on this dataset, it was shown that the proposed approach outperforms KWS systems based on other representations or a system using a sliding window. Last but not least, it was shown that exploiting temporally reversed segments of provided training samples improves the performance regardless of the embedding type. For future work, the proposed method will be evaluated in noisy conditions and for zero-shot KWS by pre-training the embeddings on a large ASR dataset as done in [20] and/or using pre-trained ASR embeddings instead of spectrograms as input representations.

5. ACKNOWLEDGMENTS

The authors would like to thank Paul M. Baggenstoss, Fabian Fritz, Lukas Henneke and Frank Kurth for their valuable feedback.

6. REFERENCES

- [1] Iván López-Espejo, Zheng-Hua Tan, John H. L. Hansen, and Jesper Jensen, “Deep spoken keyword spotting: An overview,” *IEEE Access*, vol. 10, pp. 4169–4199, 2022.
- [2] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni, “Generalizing from a few examples: A survey on few-shot learning,” *ACM Comput. Surv.*, vol. 53, no. 3, pp. 63:1–63:34, 2021.
- [3] Johan Schalkwyk, Doug Beeferman, Françoise Beaufays, Bill Byrne, Ciprian Chelba, Mike Cohen, Maryam Kamvar, and Brian Strope, ““Your word is my command”: Google search

- by voice: A case study,” *Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics*, pp. 61–90, 2010.
- [4] Assaf Hurwitz Michaely, Xuedong Zhang, Gabor Simko, Carolina Parada, and Petar S. Aleksic, “Keyword spotting for google assistant using contextual speech recognition,” in *ASRU*. 2017, pp. 272–278, IEEE.
- [5] Ami Moyal, Vered Aharonson, Ella Tetariy, and Michal Gishri, *Phonetic Search Methods for Large Speech Databases*, Springer Briefs in Electrical and Computer Engineering, Springer, 2013.
- [6] Raghav Menon, Armin Saeb, Hugh Cameron, William Kibira, John A. Quinn, and Thomas Niesler, “Radio-browsing for developmental monitoring in Uganda,” in *ICASSP*. 2017, pp. 5795–5799, IEEE.
- [7] Herman Kamper, Weiran Wang, and Karen Livescu, “Deep convolutional acoustic word embeddings using word-pair side information,” in *ICASSP*. 2016, pp. 4950–4954, IEEE.
- [8] Haoxin Ma, Ye Bai, Jiangyan Yi, and Jianhua Tao, “Hypersphere embedding and additive margin for query-by-example keyword spotting,” in *APSIPA ASC*. 2019, pp. 868–872, IEEE.
- [9] Jake Snell, Kevin Swersky, and Richard S. Zemel, “Prototypical networks for few-shot learning,” in *NeurIPS*, 2017, pp. 4077–4087.
- [10] Mark Mazumder, Colby R. Banbury, Josh Meyer, Pete Warden, and Vijay Janapa Reddi, “Few-shot keyword spotting in any language,” in *Interspeech*. 2021, pp. 4214–4218, ISCA.
- [11] Archit Parnami and Minwoo Lee, “Few-shot keyword spotting with prototypical networks,” in *ICMLT*. 2022, pp. 277–283, ACM.
- [12] Byeonggeun Kim, Seunghan Yang, Inseop Chung, and Simyung Chang, “Dummy prototypical networks for few-shot open-set keyword spotting,” in *Interspeech*. 2022, pp. 4621–4625, ISCA.
- [13] Inês Nolasco, Shubhr Singh, E. Vidiña-Villa, E. Grout, J. Morford, M. G. Emmerson, F. H. Jensen, Ivan Kiskin, H. Whitehead, Ariana Strandburg-Peshkin, Lisa F. Gill, Hanna Pamula, Vincent LOSTANLEN, Veronica Morfi, and Dan Stowell, “Few-shot bioacoustic event detection at the DCASE 2022 challenge,” in *DCASE*, 2022, pp. 136–140.
- [14] Yu Wang, Justin Salamon, Nicholas J. Bryan, and Juan Pablo Bello, “Few-shot sound event detection,” in *ICASSP*. 2020, pp. 81–85, IEEE.
- [15] Yu Wang, Mark Cartwright, and Juan Pablo Bello, “Active few-shot learning for sound event detection,” in *Interspeech*. 2022, pp. 1551–1555, ISCA.
- [16] Jinyu Li et al., “Recent advances in end-to-end automatic speech recognition,” *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, 2022.
- [17] Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *ICML*. 2006, pp. 369–376, ACM.
- [18] Byeonggeun Kim, Mingu Lee, Jinkyu Lee, Yeonseok Kim, and Kyuwoong Hwang, “Query-by-example on-device keyword spotting,” in *ASRU*. 2019, pp. 532–538, IEEE.
- [19] Li Meirong, Zhang Shaoying, Cheng Chuanxu, and Xu Wen, “Query-by-example on-device keyword spotting using convolutional recurrent neural network and connectionist temporal classification,” in *ICSP*, 2021, pp. 1291–1294.
- [20] R. Kirandevraj, Vinod Kumar Kurmi, Vinay P. Namboodiri, and C. V. Jawahar, “Generalized keyword spotting using ASR embeddings,” in *Interspeech*. 2022, pp. 126–130, ISCA.
- [21] Alan Jeffares, Qinghai Guo, Pontus Stenetorp, and Timoleon Moraitis, “Spike-inspired rank coding for fast and accurate recurrent neural networks,” in *ICLR*. 2022, OpenReview.net.
- [22] Dirk Von Zeddelmann, Frank Kurth, and Meinard Müller, “Perceptual audio features for unsupervised key-phrase detection,” in *ICASSP*. IEEE, 2010, pp. 257–260.
- [23] Raghav Menon, Herman Kamper, John A. Quinn, and Thomas Niesler, “Fast ASR-free and almost zero-resource keyword spotting using DTW and CNNs for humanitarian monitoring,” in *Interspeech*. 2018, pp. 2608–2612, ISCA.
- [24] Kevin Wilkinghoff, Alessia Cornaggia-Urrigshardt, and Fahrettin Gökgez, “Two-dimensional embeddings for low-resource keyword spotting based on dynamic time warping,” in *ITG Speech*. 2021, pp. 9–13, VDE-Verlag.
- [25] Keon Lee, Kyumin Park, and Daeyoung Kim, “DailyTalk: Spoken dialogue dataset for conversational text-to-speech,” in *ICASSP*. 2023, IEEE.
- [26] Pete Warden, “Speech commands: A dataset for limited-vocabulary speech recognition,” *CoRR*, vol. abs/1804.03209, 2018.
- [27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” in *CVPR*. 2016, pp. 770–778, IEEE.
- [28] Xiao Zhang, Rui Zhao, Yu Qiao, Xiaogang Wang, and Hongsheng Li, “AdaCos: Adaptively scaling cosine logits for effectively learning deep face representations,” in *CVPR*. 2019, pp. 10823–10832, IEEE.
- [29] Diederik P. Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015.
- [30] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz, “Mixup: Beyond empirical risk minimization,” in *ICLR*, 2018.
- [31] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le, “SpecAugment: A simple data augmentation method for automatic speech recognition,” in *Interspeech*. 2019, pp. 2613–2617, ISCA.
- [32] Frank Kurth and Dirk von Zeddelmann, “An analysis of MFCC-like parametric audio features for keyphrase spotting applications,” in *ITG Speech*. 2010, VDE.
- [33] François Petitjean, Alain Ketterlin, and Pierre Gançarski, “A global averaging method for dynamic time warping, with applications to clustering,” *Pattern recognition*, vol. 44, no. 3, pp. 678–693, 2011.
- [34] Kevin Wilkinghoff, “Sub-cluster AdaCos: Learning representations for anomalous sound detection,” in *IJCNN*. 2021, IEEE.
- [35] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen, “Metrics for polyphonic sound event detection,” *Applied Sciences*, vol. 6, no. 6, pp. 162, 2016.

A.1.9 *Key publication 9*

Kevin Wilkinghoff and Keisuke Imoto. “F1-EV Score: Measuring the Likelihood of Estimating a Good Decision Threshold for Semi-Supervised Anomaly Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 256–260. DOI: [10 . 1109 / ICASSP48485 . 2024 . 10446011](https://doi.org/10.1109/ICASSP48485.2024.10446011).

© 2024 IEEE.

The co-author of this publication contributed in the following ways: *Keisuke Imoto* provided the dataset used for the experiments. He also proposed to include Section 3.3 and Figure 5.

F1-EV SCORE: MEASURING THE LIKELIHOOD OF ESTIMATING A GOOD DECISION THRESHOLD FOR SEMI-SUPERVISED ANOMALY DETECTION

Kevin Wilkinghoff¹ and Keisuke Imoto²

¹Fraunhofer FKIE, Fraunhoferstraße 20, 53343 Wachtberg, Germany

²Doshisha University, 1-3 Tatara Miyakodani, Kyotanabe, Kyoto, Japan

kevin.wilkinghoff@ieee.org, keisuke.imoto@ieee.org

ABSTRACT

Anomalous sound detection (ASD) systems are usually compared by using threshold-independent performance measures such as AUC-ROC. However, for practical applications a decision threshold is needed to decide whether a given test sample is normal or anomalous. Estimating such a threshold is highly non-trivial in a semi-supervised setting where only normal training samples are available. In this work, F1-EV a novel threshold-independent performance measure for ASD systems that also includes the likelihood of estimating a good decision threshold is proposed and motivated using specific toy examples. In experimental evaluations, multiple performance measures are evaluated for all systems submitted to the ASD task of the DCASE Challenge 2023. It is shown that F1-EV is strongly correlated with AUC-ROC while having a significantly stronger correlation with the F1-score obtained with estimated and optimal decision thresholds than AUC-ROC.

Index Terms— performance measure, anomaly detection, decision threshold, domain generalization

1. INTRODUCTION

There are several performance measures for sound event detection [1, 2] and anomalous sound detection (ASD) systems or binary classifiers in general [3, 4, 5, 6]. Usually, threshold-independent performance measures such as the area under the receiver operating characteristic curve (AUC-ROC) are used because they are more objective [7, 8] compared to threshold-dependent measures such as the F1-score, which rely on a single chosen decision threshold. However, it has been shown that the AUC-ROC has a few weaknesses [9] as for example that AUC-ROC does not work well for imbalanced class distributions since equal weight is given for positive and negative samples [9, 10]. Moreover, for practical applications a decision threshold is needed to be able to decide whether a test sample is normal or anomalous. In semi-supervised ASD settings, estimating this threshold boils down to separating the extreme values of the anomaly scores, e.g. scores larger than the 90th percentile, belonging to normal samples from the rest and hoping that the estimated decision threshold also works well for separating the scores of normal samples from the ones obtained with anomalous samples [11]. Hence, it is implicitly assumed that the anomaly scores belonging to the normal training and test samples follow the same distribution. Estimating a good decision threshold is a difficult task that is vital when developing a system for a practical ASD application. If only AUC-ROC is used as a performance measure, this difficulty is not explicitly captured by the resulting score.

A naïve solution to this problem is to use multiple performance measures, for example a threshold-independent and a threshold-

dependent measure such as the AUC-ROC and the F1-score. However, a single threshold-independent measure is much more favorable to objectively compare the performance of multiple systems. In this work, we propose the F1-expected value (EV) score for measuring the performance of an ASD system. Similar to AUC-ROC, F1-EV is a threshold-independent performance measure but also takes the likelihood of estimating a good decision threshold into account.

The contributions of this work are the following: First, it is shown experimentally and through toy examples that AUC-ROC alone is not a sufficient performance measure for ASD. Second, F1-EV¹ a threshold-independent performance measure for anomaly detection is presented. Furthermore, multiple performance measures are evaluated and compared using all systems submitted to the ASD task of the DCASE 2023 Challenge. It is shown that bounding the F1-EV score is important and F1-EV is strongly correlated with AUC-ROC while also having a significantly higher correlation with the F1-score than AUC-ROC. As a last contribution, fine-tuning the bounds of F1-EV score is investigated in an ablation study.

2. PERFORMANCE MEASURES

Throughout this work, anomaly scores are positive scalar values, and their magnitude corresponds to the degree of anomaly.

2.1. AUC-ROC

First, we will recall the definition of the AUC-ROC score. For a set of $N \in \mathbb{N}$ threshold values $\theta(n) \in \mathbb{R}$ indexed by $n \in \{1, \dots, N\}$, let $x(n), y(n) \in [0, 1]$ be the monotone increasing sequences of sorted false positive rates (FPRs) and sorted true positive rates (TPRs) of the receiver operating characteristic (ROC)-curve resulting from the anomaly scores, respectively. Define

$$\Delta x(n) := x(n+1) - x(n).$$

More concretely, the sequence $(\theta(n))_{n=1, \dots, N}$ is set to all sorted anomaly score values belonging to the evaluation samples because these are the points where the intermediate statistics, i.e. TPR and FPR, change. Compared to linearly scaled thresholds, which require an infinitesimal resolution to yield exact results, this particular choice of evaluation thresholds has two advantages [8]: improved computational efficiency and improved accuracy of the ROC curve. Then, using the trapezoidal rule, the AUC-ROC score can be approximated by calculating

$$\text{AUC-ROC}(x, y) \approx \sum_{n=1}^{N-1} \frac{y(n+1) + y(n)}{2} \cdot \Delta x(n)$$

¹An open-source implementation of F1-EV is available at: <https://github.com/wilkinghoff/f1-ev>

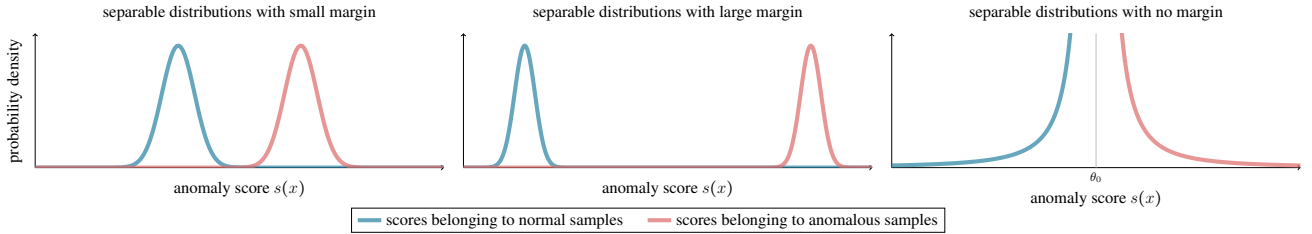


Fig. 1: Three toy examples of perfectly separable anomaly score distributions, each with an AUC-ROC equal to 1. On the left, the margin between normal and anomalous scores is small. In the center, the margin between the distributions is large and thus estimating a good decision threshold is much easier. On the right, the only optimal decision threshold is $\theta = \theta_0$, which is a null set with measure zero, and thus estimating this threshold also has a likelihood of zero when assuming a continuous distribution with uncountable support for estimated thresholds. Note that when estimating a decision threshold, only a finite number of anomaly scores belonging to the distribution of normal samples are available and usually both distributions are more complex and overlap. This is the reason why estimating a good decision threshold is highly non-trivial.

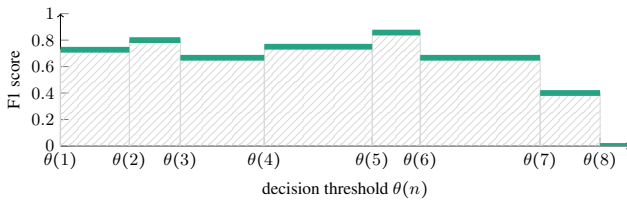


Fig. 2: Example of computing F1-EV using only very few decision thresholds for illustration purposes.

as implemented in [12].

AUC-ROC does not require a specifically chosen decision threshold. This allows for objective comparisons between different ASD systems but does not take into account the difficulty of estimating a good decision threshold. The AUC-ROC score is equal to the probability that the anomaly score of a random normal sample is smaller than the anomaly score of a random anomalous sample [7, 13]. Hence, even when AUC-ROC equals 1, the results can look very different because such a value only shows that there exists a threshold that perfectly separates the anomaly scores belonging to normal and anomalous samples as illustrated in Figure 1. This motivates the definition of other threshold-independent performance measures as proposed in the following two subsections.

2.2. F1-EV

The idea of F1-EV is to calculate the expected value of a random variable that models the F1-score of an anomaly detection system. This is realized by calculating the F1-score for all possible threshold values $\theta(n)$ and computing the area under the resulting piecewise constant F1-score function. In [14], it has been shown that computing the AUC-ROC of the precision-recall curve by using linear interpolations between samples as done by the trapezoidal rule leads to over-optimistic results. Since the F1-score is the harmonic mean of precision and recall, it thus makes more sense to also not use the trapezoidal rule. We utilize a finite Riemann sum instead as illustrated in Figure 2. Since the empirical F1-score function is piece-wise constant with respect to the decision threshold, using more threshold values than the number of samples does not improve the accuracy of the Riemann sum.

Let us now formally introduce F1-EV. Define the normalized

distance between thresholds as

$$\Delta\theta(n) := \frac{\theta(n+1) - \theta(n)}{\theta(N) - \theta(1)}.$$

Further, let $F1(\theta(n))$ denote the F1-score obtained when applying the decision threshold $\theta(n)$. Then, the F1-EV score is defined as

$$F1-EV(\theta) := \sum_{n=1}^{N-1} F1(\theta(n)) \cdot \Delta\theta(n).$$

When assuming a uniform distribution for estimating a threshold in the interval $[\theta(1), \theta(N)]$, the value $\Delta\theta(n)$ corresponds to the likelihood of estimating the decision threshold $\theta(n)$. In conclusion, F1-EV is the expected value of a random variable that models the obtained F1-score of an anomaly detection system, as intended, and thus yields a score between 0 and 1 with higher values indicating better performance.

2.3. Bounded F1-EV

When setting the range of possible thresholds, one has to define a smallest and a largest value of possible thresholds to compute a performance measure. For F1-EV, these values have been set to the smallest and largest anomaly scores of a given set of test samples. However, many of these thresholds are very unlikely to be estimated. One can expect that estimated decision thresholds lie in a certain range that depends on the estimation method. Furthermore, if there are a few outliers in the anomaly score samples that are much smaller or much larger than all the other scores, this will skew the final performance measure drastically. Therefore, we propose to define more robust boundaries for the interval of possible thresholds by only utilizing the anomaly scores belonging to the normal test samples. Hence, the evaluation is independent of the particular set of normal training samples, which are assumed to follow the same distribution as the validation and test samples when estimating a decision threshold. Furthermore, in [11] it has been shown that holding back a few normal training samples when training the system to obtain more realistic anomaly scores from these samples for estimating a decision threshold does not increase the resulting F1-score. This indicates that the assumption made about the distributions is valid.

We set the lower bound of the decision threshold, denoted by $\theta_{\min} \in \mathbb{R}$, to be close to the sample mean μ of the anomaly scores belonging to the normal test samples. Here we assume that, ideally, one would not estimate a decision threshold that is much smaller than this value. An upper bound $\theta_{\max} \in \mathbb{R}$ is more difficult to choose

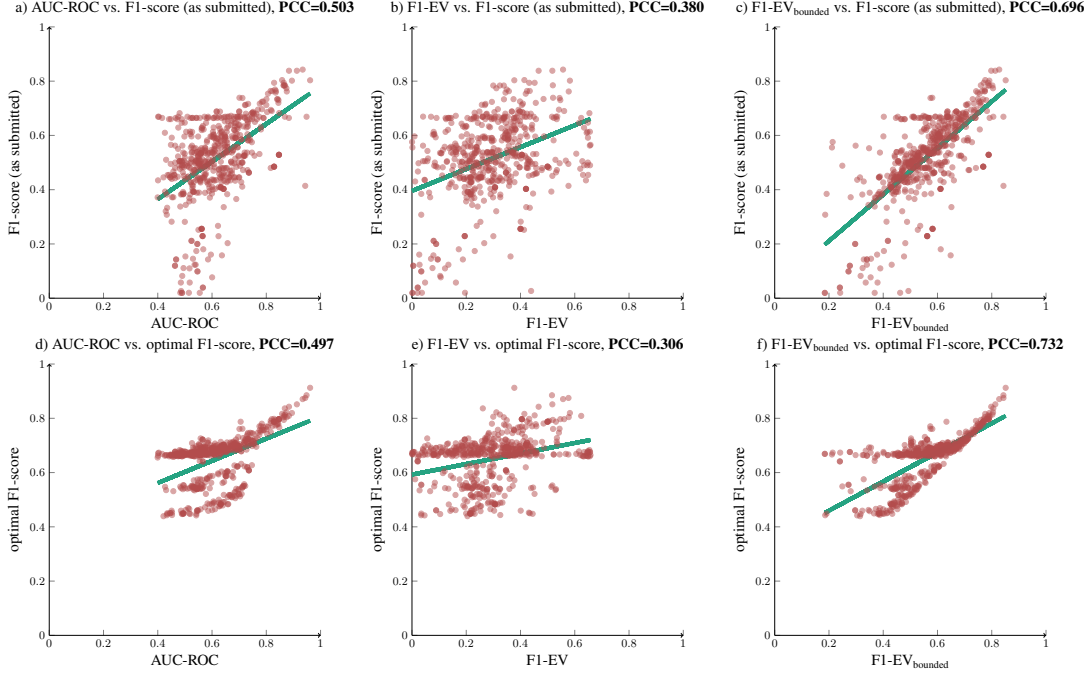


Fig. 3: Comparison of several different performance measures computed on the evaluation set of task 2 of the DCASE2023 Challenge. In the top row, threshold-independent performance measures are compared to the F1-score obtained with the submitted decision threshold. In the bottom row, threshold-independent performance measures are compared to the F1-score obtained with an optimal decision threshold.

because the largest value of the anomaly scores may be an outlier and $+\infty$ is definitely too large. Instead, we propose to utilize an upper bound close to the empirically optimal decision threshold $\theta_{\text{opt}} \in \mathbb{R}$, which is chosen as the center of the interval of possible thresholds yielding optimal results. Again, the underlying assumption is that, in an ideal world, one would not estimate a threshold much larger than this upper bound. More concretely, we set

$$\theta_{\text{max}} := \theta_{\text{opt}} + \alpha \cdot \sigma \text{ and } \theta_{\text{min}} := \mu - \alpha \cdot \sigma$$

where $\sigma \in \mathbb{R}$ denotes the sample standard deviation belonging to the normal test samples and $\alpha \in \mathbb{R}$ is a hyperparameter to be set. For the experiments conducted in this work, we manually set $\alpha = 0.2$. More details on choosing an appropriate value for α can be found in subsection 3.3. Now, for $k = 1, \dots, K$ with $K = 2 + |\{n \in \{1, \dots, N\} : \theta_{\text{min}} < \theta(n) < \theta_{\text{max}}\}|$ we set

$$\theta_{\text{bounded}}(k) = \begin{cases} \theta_{\text{min}} & \text{if } k = 1 \\ \theta(k-1) & \text{if } \theta_{\text{min}} < \theta(k-1) < \theta_{\text{max}} \\ \theta_{\text{max}} & \text{if } k = K \end{cases} .$$

The normalized distances between thresholds are adapted from the distances used for the basic F1-EV score

$$\Delta\theta_{\text{bounded}}(k) := \frac{\theta_{\text{bounded}}(k+1) - \theta_{\text{bounded}}(k)}{\theta_{\text{bounded}}(K) - \theta_{\text{bounded}}(1)}$$

and the bounded F1-EV score is set to

$$\text{F1-EV}_{\text{bounded}}(\theta_{\text{bounded}}) := \sum_{k=1}^{K-1} \text{F1}(\theta_{\text{bounded}}(k)) \cdot \Delta\theta_{\text{bounded}}(k).$$

Again, this results in a score between 0 and 1 with higher values indicating better performance.

3. EXPERIMENTAL EVALUATIONS

3.1. Experimental setup

To compare different performance measures, we computed Pearson correlation coefficients (PCCs) using all systems submitted to the ASD task of the DCASE2023 Challenge for machine condition monitoring [15, 16, 17, 18]. This dataset consists of noisy audio recordings with a length between 6 and 18 seconds belonging to 14 different machine types that are split into a development set and an evaluation set. For each machine type contained in one of the sets, there is a source domain with 990 normal training samples and a target domain with 10 normal training samples that differs from the source domain by changing machine parameters or the background noise. The test splits of the development and evaluation set consist of 100 and 200 samples for each machine type, respectively. Any test sample is either normal or anomalous and belongs to the source or target domain. The task is to predict whether a given test sample is normal or anomalous by using a single decision threshold for each machine type regardless of the domain the sample belongs to (domain generalization [19]). To determine the ranking of the systems in the challenge, the harmonic mean of AUC-ROC and partial AUC-ROC [20] over all machine types was used. Therefore, estimating a proper decision threshold to obtain a good F1-score is only optional and some participants abstained from doing this. To not skew the results in this work, we only included the submissions with an F1-score greater than 0 for the experimental evaluations. Furthermore, we used the performance measures of all submitted systems for each machine type independently because another decision threshold is used for each machine type.

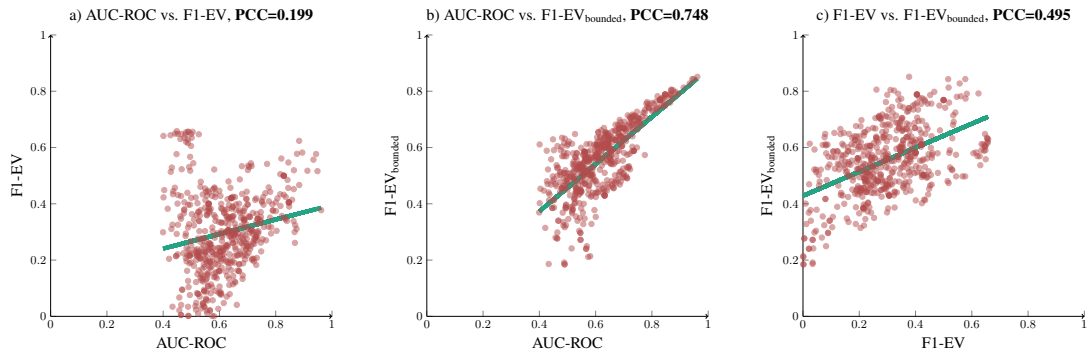


Fig. 4: Comparison of threshold-independent performance measures computed on the evaluation set of task 2 of the DCASE2023 Challenge.

3.2. Comparison of performance measures

The experimental results are depicted in Figure 3 and Figure 4. The following observations can be made: First, AUC-ROC has only a low to moderate correlation with the F1-scores belonging to the estimated decision thresholds ($PCC=0.503$) and the optimal F1-scores ($PCC=0.497$), which experimentally justifies the motivation for F1-EV as a performance measure. Second, the basic F1-EV score has only very low correlation with AUC-ROC ($PCC=0.199$) and also low correlation with the F1-scores ($PCC=0.380$ and $PCC=0.306$) showing that it is necessary to define proper bounds. Third and most importantly, the bounded F1-EV score, has a strong correlation with AUC-ROC ($PCC=0.748$) and both F1-scores ($PCC=0.696$ and $PCC=0.732$). Hence, F1-EV better incorporates the difficulty of finding a good decision threshold than AUC-ROC and thus may also be a more suitable performance measure for ASD. Note that some of the submissions may have estimated improper decision thresholds and thus the correlation between the bounded F1-EV score and the F1-scores belonging to the estimated decision thresholds may be higher when only suitable estimation techniques are applied.

The clusters shaped as horizontal lines in Figure 3 at an F1-score of approximately two-thirds, for example ranging from an AUC-ROC of 0.4 to 0.9 in Subfig. a), look odd but can be explained as follows: It is possible that a chosen decision threshold works very well for the source domain but particularly bad for the target domain or vice versa. This means that precision or recall is close to 1 for both domains and the other value is close to 0 for one domain and close to 1 for the other domain, thus in total close to 0.5 for both domains assuming that both domains have approximately the same number of samples. Since the F1-score is the harmonic mean of precision and recall, this results in an F1-score of approximately two-thirds.

3.3. Choosing the hyperparameter α

As an ablation study, the sensitivity of the bounded F1-EV score with respect to the parameter α was investigated. The results, depicted in Figure 5, show that the PCC between the bounded F1-EV and AUC-ROC decreases for $\alpha > 0.2$ showing that both measures are indeed different. Furthermore, the PCC between the bounded F1-EV and the optimal F1-score increases for $\alpha < 1$ and slightly decreases for $\alpha > 1$ but keeps having a high correlation. Lastly and most importantly, the PCC between the bounded F1-EV and the F1-score obtained with the submitted decision thresholds slightly increases for $\alpha < 0.2$ and slightly decreases for $\alpha > 0.2$ but to much less degree as it is the case for AUC-ROC. In conclusion, the bounded F1-EV is relatively stable with respect to α when considering estimated

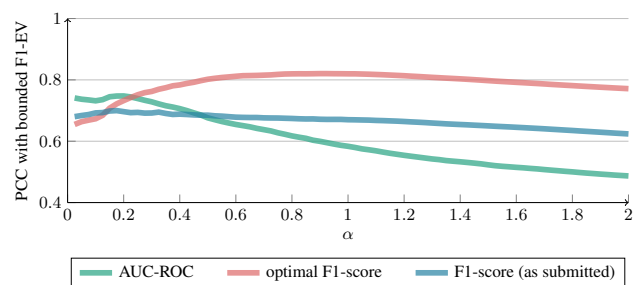


Fig. 5: Sensitivity of the bounded F1-EV score with respect to α .

decision thresholds. As the purpose of F1-EV is to have a threshold-independent performance measure that is similar to AUC-ROC but also incorporates the difficulty of estimating a good decision threshold, we propose to use a relatively small value for α , e.g. $\alpha = 0.2$, as done in the other experiments of this work.

4. CONCLUSION

In this work, the threshold-independent performance measure F1-EV for ASD systems that combines the advantages of both the threshold-independent AUC-ROC and threshold-dependent F1-score by correlating well with both of them was proposed. In experiments conducted on the predictions of all systems submitted to the DCASE2023 ASD Challenge, it was shown that a bounded F1-EV has a strong correlation with AUC-ROC while having a much higher correlation with the F1-scores based on estimated and optimal decision thresholds. In conclusion, this performance measure has the potential to replace AUC-ROC as the de facto standard performance measure for ASD. For future work, it is planned to conduct additional experiments on other datasets and further optimize the upper and lower bound of F1-EV. In particular, choosing other distributions than a uniform distribution for estimating a decision threshold in the allowed range may be a promising direction. Other experiments may focus on evaluating F1-EV for settings with a strong imbalance between normal and anomalous samples and investigate specific machine types or domains individually.

5. ACKNOWLEDGMENTS

The authors would like to thank Fabian Fritz and Frank Kurth for their valuable feedback.

6. REFERENCES

- [1] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen, “Metrics for polyphonic sound event detection,” *Applied Sciences*, vol. 6, no. 6, pp. 1–17, 2016.
- [2] Giacomo Ferroni, Nicolas Turpault, Juan Azcarreta, Francesco Tuveri, Romain Serizel, Çağdas Bilen, and Sacha Krstulovic, “Improving sound event detection metrics: Insights from DCASE 2020,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2021, pp. 631–635, IEEE.
- [3] Oluwasanmi Koyejo, Nagarajan Natarajan, Pradeep Ravikumar, and Inderjit S. Dhillon, “Consistent binary classification with generalized performance metrics,” in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2014, pp. 2744–2752.
- [4] John Paparrizos, Paul Boniol, Themis Palpanas, Ruey Tsay, Aaron J. Elmore, and Michael J. Franklin, “Volume under the surface: A new accuracy evaluation measure for time-series anomaly detection,” *Proc. VLDB Endow.*, vol. 15, no. 11, pp. 2774–2787, 2022.
- [5] Kendrick Boyd, Kevin H. Eng, and C. David Page Jr., “Area under the precision-recall curve: Point estimates and confidence intervals,” in *Machine Learning and Knowledge Discovery in Databases - European Conference (ECML/PKDD)*. 2013, vol. 8190 of *Lecture Notes in Computer Science*, pp. 451–466, Springer.
- [6] Gurol Canbek, Tugba Taskaya-Temizel, and Seref Sagiroglu, “PToPI: A comprehensive review, analysis, and knowledge representation of binary classification performance measures/metrics,” *SN Comput. Sci.*, vol. 4, no. 1, pp. 1–30, 2023.
- [7] Charu Aggarwal, *Outlier Analysis*, Springer, 2nd edition, 2017.
- [8] Janek Ebbers, Reinhold Haeb-Umbach, and Romain Serizel, “Threshold independent evaluation of sound event detection scores,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 1021–1025, IEEE.
- [9] Jorge M Lobo, Alberto Jiménez-Valverde, and Raimundo Real, “AUC: A misleading measure of the performance of predictive distribution models,” *Global ecology and Biogeography*, vol. 17, no. 2, pp. 145–151, 2008.
- [10] Takaya Saito and Marc Rehmsmeier, “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets,” *PLoS one*, vol. 10, no. 3, e0118432, 2015.
- [11] Kevin Wilkinghoff and Alessia Cornaggia-Urrigshardt, “On choosing decision thresholds for anomalous sound detection in machine condition monitoring,” in *24th International Congress on Acoustics (ICA)*. 2022, The Acoustical Society of Korea.
- [12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [13] James A. Hanley and Barbara J. McNeil, “The meaning and use of the area under a receiver operating characteristic (roc) curve,” *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.
- [14] Jesse Davis and Mark Goadrich, “The relationship between precision-recall and ROC curves,” in *Twenty-Third International Conference on Machine Learning (ICML)*. 2006, vol. 148 of *ACM International Conference Proceeding Series*, pp. 233–240, ACM.
- [15] Noboru Harada, Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Masahiro Yasuda, and Shoichiro Saito, “Toy-ADMOS2: Another dataset of miniature-machine operating sounds for anomalous sound detection under domain shift conditions,” in *6th Workshop on Detection and Classification of Acoustic Scenes and Events*, 2021, pp. 1–5.
- [16] Noboru Harada, Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, and Masahiro Yasuda, “First-shot anomaly detection for machine condition monitoring: A domain generalization baseline,” in *31st European Signal Processing Conference (EUSIPCO)*. 2023, pp. 191–195, IEEE.
- [17] Kota Dohi, Tomoya Nishida, Harsh Purohit, Ryo Tanabe, Takashi Endo, Masaaki Yamamoto, Yuki Nikaido, and Yohei Kawaguchi, “MIMII DG: sound dataset for malfunctioning industrial machine investigation and inspection for domain generalization task,” in *7th Workshop on Detection and Classification of Acoustic Scenes and Events*, 2022, pp. 26–30.
- [18] Kota Dohi, Keisuke Imoto, Noboru Harada, Daisuke Niizumi, Yuma Koizumi, Tomoya Nishida, Harsh Purohit, Ryo Tanabe, Takashi Endo, and Yohei Kawaguchi, “Description and discussion on DCASE 2023 challenge task 2: First-shot unsupervised anomalous sound detection for machine condition monitoring,” in *8th Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2023, pp. 31–35.
- [19] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, and Tao Qin, “Generalizing to unseen domains: A survey on domain generalization,” in *Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*. 2021, pp. 4627–4635, ijcai.org.
- [20] Donna Katzman McClish, “Analyzing a portion of the ROC curve,” *Medical decision making*, vol. 9, no. 3, pp. 190–195, 1989.



A.1.10 *Key publication 10*

Kevin Wilkinghoff and Frank Kurth. “Why do Angular Margin Losses work well for Semi-Supervised Anomalous Sound Detection?” In: *IEEE/ACM Transactions on Audio, Speech and Language Processing* 32 (2024), pp. 608–622. DOI: [10.1109/TASLP.2023.3337153](https://doi.org/10.1109/TASLP.2023.3337153).

© 2024 IEEE (CC-BY license).

The co-author of this publication contributed in the following ways: *Frank Kurth* contributed through scientific supervision and improving the quality of the paper in general. He also proposed to include Table III and Figure 1.

Why do Angular Margin Losses work well for Semi-Supervised Anomalous Sound Detection?

Kevin Wilkinghoff , *Student Member, IEEE*, and Frank Kurth , *Senior Member, IEEE*

Abstract—State-of-the-art anomalous sound detection systems often utilize angular margin losses to learn suitable representations of acoustic data using an auxiliary task, which usually is a supervised or self-supervised classification task. The underlying idea is that, in order to solve this auxiliary task, specific information about normal data needs to be captured in the learned representations and that this information is also sufficient to differentiate between normal and anomalous samples. Especially in noisy conditions, discriminative models based on angular margin losses tend to significantly outperform systems based on generative or one-class models. The goal of this work is to investigate why using angular margin losses with auxiliary tasks works well for detecting anomalous sounds. To this end, it is shown, both theoretically and experimentally, that minimizing angular margin losses also minimizes compactness loss while inherently preventing learning trivial solutions. Furthermore, multiple experiments are conducted to show that using a related classification task as an auxiliary task teaches the model to learn representations suitable for detecting anomalous sounds in noisy conditions. Among these experiments are performance evaluations, visualizing the embedding space with t-SNE and visualizing the input representations with respect to the anomaly score using randomized input sampling for explanation.

Index Terms—representation learning, anomaly detection, angular margin loss, compactness loss, machine listening, domain generalization, explainable artificial intelligence

I. INTRODUCTION

SEMI-SUPERVISED anomalous sound detection (ASD) is the task of reliably detecting anomalous sounds while only having access to normal sounds for training a model [1]. Since anomalies occur only rarely by definition and usually are very diverse, collecting realistic anomalous samples for training a system is much more difficult and thus more costly than collecting normal data. Hence, a semi-supervised ASD setting is more realistic than a supervised ASD setting, for which anomalous sounds are available for training, because it substantially simplifies the data collection process. There are also unsupervised ASD settings, for which the training dataset may also contain anomalous samples and it is unknown whether a training sample is normal or anomalous. But for many applications, it can be ensured that only normal samples are collected for training and thus a semi-supervised setting can be assumed.

The authors are with Fraunhofer FKIE, Fraunhoferstraße 20, 53343 Wachtberg, Germany (e-mail: kevin.wilkinghoff@ieee.org, frank.kurth@fkie.fraunhofer.de).

Manuscript received March 27, 2023; revised September 19, 2023 and November 20, 2023, accepted for publication November 23, 2023.

ASD has many applications. Examples are machine condition monitoring [2]–[4], medical diagnosis [5], [6], bioacoustic monitoring [7], [8], intrusion detection in smart home environments [9] and detecting crimes [10], [11] or accidents [12], [13]. Furthermore, detecting anomalous samples can also be understood as a subtask in acoustic open-set classification [14]–[16]. Throughout this work, we will use machine condition monitoring in domain-shifted conditions as an application example [4]. Here, the audio signals may contain one or several of the following three components: 1) normal machine sounds, 2) anomalous machine sounds and 3) background noise consisting of a mixture of many other sound events. The major difficulty of this ASD application is that anomalous components of machine sounds can be very subtle when being compared to the background noise making it difficult to reliably detect anomalous signal components. Furthermore, machine sounds and background noise can change substantially for different domain shifts, which we define as alterations in the (acoustic) environment or changes in parameter settings of the machines. The ASD system still needs to only detect anomalous signal components without frequently raising false alarms caused by any domain shift.

There are several strategies to train an ASD system for machine condition monitoring using only normal data. Among these strategies are generative models such as autoencoders [17]–[22] or normalizing flows [23], [24] that directly try to model the probability distribution of normal data, which is also called inlier modeling (IM) [3]. Another strategy is to use an auxiliary task, usually a classification task, for training a model to learn meaningful representations of the data (embeddings) that can be used to identify anomalies. Possible auxiliary tasks for machine condition monitoring are classifying between machine types [25]–[29] or, additionally, between different machine states and noise settings [30]–[33], recognizing augmented and not augmented versions of normal data (self-supervised learning) [25] or predicting the activity of machines [32]. Using an auxiliary task to learn embeddings is also called outlier exposure (OE) [34] because normal samples belonging to other classes than a target class can be considered as proxy outliers [35]. Often an angular margin loss such as SphereFace [36], CosFace [37] or ArcFace [38] is utilized for training an OE model. Systems based on embeddings pre-trained on very large datasets [39]–[41] can be used, too. However, it has been shown that directly training a system on the data yields better ASD results, even when only very limited training data is available [42]. In addition, different

strategies can be combined by using an ensemble of multiple models [43]–[45].

Different strategies to train an ASD system have different strengths and weaknesses. Using an auxiliary task for training relies on additional meta-information to generate labels for a classification task whereas IM-based models do not need any labels. Furthermore, autoencoders can localize anomalies in the input space by visualizing an element-wise reconstruction error as done in [19], [21]. However, training ASD models by using an auxiliary task usually enhances their performance [46]. Even for IM-based models, performance can be significantly improved when utilizing meta information such as machine types. In [21] a class-conditioned autoencoder is used, in [44] not only spectral features but also the machine ID is encoded and decoded, and in [23] a normalizing flow is trained to assign lower likelihood to sounds of other machines and a higher likelihood to sounds of the target machine. As suspected in [32], [33], the most likely reason for the difference in performance is that, as stated before, recordings for machine condition monitoring are very noisy because of factory background noise. This is a problem for IM-based models because they cannot tell the difference between arbitrary sound events not emitted by a monitored machine and normal or anomalous sounds emitted by the machine. Both are considered equally important by the model. Moreover, anomalies present in these noisy audio recordings are usually very subtle when being compared to the noise or other sound events present in a recording making it even more difficult to detect potential anomalies. When being trained with an auxiliary task, a model learns to ignore noise, which can be assumed to be similar for all considered classes, and therefore to isolate the target machine sound by ignoring the uninformative background sound events. As a result, these models are more sensitive to changes of the machine sounds and have better anomaly detection capabilities.

Localizing and visualizing frequencies or temporal regions of recordings that are being considered anomalous is important for practical applications because users can better understand the decisions of the ASD system (explainable artificial intelligence (xAI) [47]). Furthermore, this may help to find the cause of mechanical failure and thus can simplify the maintenance process. As stated before, autoencoders can easily localize anomalies by using an element-wise reconstruction error. Additional investigations on visualizing and explaining ASD decisions include showing that decisions of ASD systems for machine condition monitoring largely rely on high-frequency information [48]. This has been visualized using local interpretable model-agnostic explanations (LIME) [49] applied to sounds (SLIME) [50]. Furthermore, uniform manifold approximation and projection (UMAP) [51] has been used to visualize representations of the data such as stacked consecutive frames of log magnitude spectrograms, log-mel magnitude spectrograms, or openL3 embeddings [46].

The goal of this work is to explain why angular margin losses work well for anomalous sound detection. To achieve this goal, the following contributions are made: First and foremost, it is theoretically proven that, after normalizing the embedding space, training an ASD model by minimizing an

angular margin loss using an auxiliary task can be considered as minimizing a regularized one-class loss while being less affected by noise or non-target sound events present in the data. Moreover, it is experimentally verified that using an angular margin loss for training a model to discriminate between classes of an auxiliary task also leads to better ASD performance and thus is a better choice for an ASD task than minimizing a one-class loss such as an intra-class (IC) compactness loss with a single or multiple classes. Last but not least, a procedure for visualizing normal and anomalous regions of the input representations based on randomized input sampling for explanation (RISE) is presented. Using these visualizations, it is shown that normal and anomalous sounds cannot be distinguished from the highly complex background noise when training with a one-class loss. In contrast, when using an auxiliary task with multiple classes the model learns to ignore noise and isolate the targeted machine sound for monitoring their condition.

The paper is structured as follows: In Section II, various one-class losses and angular margin losses are reviewed. Section III presents our main theoretical results about the relation between these loss functions. Section IV contains a description of the experimental setup and all experimental evaluations consisting of performance evaluations, a comparison between losses during training, visualizing normal and anomalous regions of input representations as perceived by the system and visualizing the resulting embedding spaces. Section V consists of the conclusions of this work.

II. LOSS FUNCTIONS

In this section, a unified presentation and discussion of several loss functions that are needed for presenting one of the main results of this work in Sec. III will be given. The following notation will be used throughout the paper: X denotes the space of input data samples, $N \in \mathbb{N}$ the number of classes defined for an auxiliary task and $D \in \mathbb{N}$ the dimension of the embedding space.

A. One-Class Classification Losses

When training a model for ASD while only having access to normal data i.e. a single class, this is referred to as *one-class classification* and is some form of IM. The compactness loss [52], whose goal it is to project the data into a hypersphere of minimum volume, will serve as a representative of losses for one-class classification and is defined as follows.

Definition 1 (Compactness loss). Let $Y \subset X$ be finite. Let \mathcal{P} denote the power set, Φ denote the space of network architectures for extracting embeddings and $W(\phi)$ denote the parameter space of $\phi \in \Phi$, i.e. $\phi : X \times W(\phi) \rightarrow \mathbb{R}^D$. Then, the *compactness loss* is defined as

$$\begin{aligned} \mathcal{L}_{\text{comp}} &: \mathcal{P}(X) \times \mathbb{R}^D \times \Phi \times W \rightarrow \mathbb{R}_+ \\ \mathcal{L}_{\text{comp}}(Y, c, \phi, w) &:= \frac{1}{|Y|} \sum_{x \in Y} \|\phi(x, w) - c\|_2^2. \end{aligned} \quad (1)$$

The vector $c \in \mathbb{R}^D$ is called *center*.

After training, the (squared) Euclidean distance between the embedding of a given sample and the center can be utilized as an anomaly score: A greater distance indicates a higher likelihood for the sample to be anomalous. A *trivial solution* for minimizing the compactness loss with center $c \in \mathbb{R}^D$ is a parameter setting $w_c \in W(\phi)$ such that ϕ is the constant function $\phi(x, w_c) = c$ for all $x \in X$. It is of utmost importance to prevent that the model to be trained is able to learn such a trivial solution. Otherwise it is impossible to differentiate between normal and anomalous samples.

There are several strategies to prevent a model from learning a trivial solution. First of all, it needs to be ensured that $c \neq c_0 \in \mathbb{R}^D$ where $c_0 = \phi(x, w_0)$ is defined as the output of the network obtained by setting the weight parameters of model ϕ to zero. This is because we have $\phi(x, w_0) = c_0$ for all $x \in X$ as long as the model uses only linear operators, e.g. dense or convolutional layers, and all activation functions have zero as a fixed point, which is the case for most commonly used activation functions. In [52], it has been shown that using bias terms, bounded activation functions or a trainable center all enable the model to learn a constant function when using an additive weight decay regularization term and thus must also be avoided.

Another possibility to avoid trivial solutions is to impose additional tasks, so-called *auxiliary tasks*, not directly related to the ASD problem while training. Autoencoders [53], which are trained to first encode and then decode the input again and have many interesting applications by themselves such as denoising data [54], can also be viewed as a way to regularize one-class models. Here, the encoder is the one-class model mapping the input to an embedding space. Learning a constant function is not a (trivial) solution for the task because all necessary information for being able to completely reconstruct the input needs to be encoded. However, noise including other sound sources present in the input audio data needs to be encoded as well because otherwise the input cannot be reconstructed. Therefore, the noise heavily influences the embeddings and thus the embeddings can also be considered noisy. Depending on the complexity of the noise, most information contained in the embeddings is only related to the noise and not to the target sound to be analyzed and thus detecting anomalies using an autoencoder may be difficult. Moreover, in [52] it has been shown that using compactness loss, even for clean datasets, outperforms commonly used autoencoder architectures when detecting anomalies.

A second choice of an auxiliary task to prevent the model from learning a constant function as a trivial solution is a classification task. Defining multiple classes through an auxiliary task inherently prevents learning a constant function as this would not be a (trivial) solution to the imposed classification problem. In [55], an additional *descriptiveness loss* is used whose goal is to reduce inter-class similarity between classes of an arbitrary, external multi-class dataset, which is only used to regularize the one-class classification task. This is done by minimizing the standard categorical cross-entropy (CCE) loss for classification on this additional dataset as an auxiliary task. For each of the two tasks, another version of the same network with identical structure and tied weights is

used. During training, both losses are jointly minimized using a weighted sum ensuring that the so-called reference network associated with the compactness loss does not learn a constant function because this would prevent the secondary network to be able to classify correctly.

Remark. The original definition of the compactness loss [52] also includes an additional weight decay term. Such a weight decay term can be used to complement any loss function and does not prevent the model from learning trivial solutions as it is still possible that the model learns to map everything to the center. Furthermore, all theoretical results presented in this work are valid regardless of whether this specific weight decay term is included or not. The proof of the main theorem can easily be modified to including the same weight decay term because it is just an additional additive term. Therefore, we omitted this term in the theoretical investigations of this work for the sake of simplicity while still using it in our experiments. However, we did not notice any significant effect on the performance.

For the remainder of this work, we propose to normalize all representations in the embedding space \mathbb{R}^D , meaning that $\|c\|_2 = 1 = \|\phi(x, w)\|_2$ for all $x \in X, w \in W(\phi)$ and centers $c \in \mathbb{R}^D$. This can easily be achieved by dividing the embeddings by their corresponding Euclidean norms. A normalization of the embedding space essentially reduces the dimension by one as evident by using stereographic projection. But doing so does not degrade the ASD performance because the dimension of the embedding space usually is larger than it needs to be.

Normalizing the embedding space has several advantages. Most importantly, the initialization of the centers is substantially simplified. In high-dimensional vector spaces i.i.d. random elements are almost surely approximately orthogonal [56]. Hence, all class centers can be randomly initialized by sampling from a uniform random distribution as also done in [33] and a careful strategy for initializing the class centers is not needed. This does not cause any problems e.g. by accidentally using class centers that are very similar to each other in terms of cosine similarity whereas the corresponding acoustic classes are very dissimilar or vice versa. Moreover, normalizing the centers ensures that all centers are distributed equidistantly and sufficiently far away from zero to avoid learning a trivial solution. Last but not least, normalizing the embeddings may even prevent numerical issues while training similar to when using batch normalization [57].

B. Angular Margin Losses

We will review the definition of ArcFace [38] as a representative of angular margin losses.

Definition 2 (ArcFace). Let $Y \subset X$ be finite and $l_j(x) \in \{0, 1\}$ denote the j th component of the categorical class label function $l \in L$ where L denotes the space of all functions $l : X \rightarrow \{0, 1\}^N$ with $\sum_{j=1}^N l_j(x) = 1$ for all $x \in X$. Let \mathcal{P} denote the power set, Φ denote the space of network architectures for extracting embeddings and $W(\phi)$ de-

note the parameter space of $\phi \in \Phi$, thus $\phi : X \times W(\phi) \rightarrow \mathbb{R}^D$. Let $\text{smax} : \mathbb{R}^N \rightarrow [0, 1]^N$ denote the softmax function, i.e.

$$\text{smax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^N \exp(x_j)}. \quad (2)$$

Then, the *ArcFace* loss is defined as

$$\begin{aligned} \mathcal{L}_{\text{ang}} : \mathcal{P}(X) \times \mathcal{P}(\mathbb{R}^D) \times \Phi \times W \times L \times \mathbb{R}_+ \times [0, \frac{\pi}{2}] &\rightarrow \mathbb{R}_+ \\ \mathcal{L}_{\text{ang}}(Y, C, \phi, w, l, s, m) \\ := - \frac{1}{|Y|} \sum_{x \in Y} \sum_{j=1}^N l_j(x) \log(\text{smax}(s \cdot \cos_{\text{mar}}(\phi(x, w), c_j, m))) \end{aligned} \quad (3)$$

where $|C| = N$ and, in this case,

$$\begin{aligned} \text{smax}(s \cdot \cos_{\text{mar}}(\phi(x, w), c_i, m)) \\ := \frac{\exp(s \cdot \cos_{\text{mar}}(\phi(x, w), c_i, m))}{\sum_{j=1}^N \exp(s \cdot \cos_{\text{mar}}(\phi(x, w), c_j, m) \cdot l_j(x))} \end{aligned} \quad (4)$$

with

$$\cos_{\text{mar}}(x, y, m) := \cos(\arccos(\cos(x, y)) + m) \quad (5)$$

for cosine similarity

$$\cos(x, y) := \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2} \in [-1, 1]. \quad (6)$$

The vectors $c_j \in \mathbb{R}^D$ are called *class centers*, $m \in [0, \frac{\pi}{2}]$ is called *margin* and $s \in \mathbb{R}_+$ is called *scale parameter*.

Remark. When using mixup [58] for data augmentation, the definition of the class label function needs to be generalized to $l : X \rightarrow [0, 1]^N$ with $\sum_{j=1}^N l_j(x) = 1$ for all $x \in X$. In the experimental evaluations of this work, mixup will be used when training a model as this improves the ASD performance [29]. Furthermore, the theoretical results presented in this work still hold when using mixup but in the proofs only binary labels will be used for the sake of simplicity.

In [59], it has been shown that the choice of both hyperparameters, the scale parameter s and the margin m , can have a significant impact on the resulting performance. Strongly varying the magnitude of one of the individual parameters has a similar effect on the sensitivity of the posterior probabilities with respect to the angles as varying the other parameter. Both a scale parameter that is too large and a margin that is too small lead to very high posterior probabilities for the target class, approximately equal to one, even for relatively large angles. Therefore, the loss function is insensitive to changing the angle. A scale parameter that is too small limits the maximum posterior probability of the target class that can be achieved. Similarly, a margin that is too large also leads to relatively small posterior probabilities. Thus, in both cases the model still tries to adapt its parameters even when the angles are already small, which hinders convergence. Due to the similar behavior of both parameters, a single appropriately chosen parameter is sufficient for controlling the posterior probabilities and it has even been shown that an adaptive scale parameter outperforms using two tuned but fixed parameters. Therefore, we will assume that s is adaptive as specified for the AdaCos loss

in [59] and set $m = 0$, i.e. $\cos_{\text{mar}}(x, y, 0) = \cos(x, y)$ for the remainder of this work. Formally, the definition of the AdaCos loss is the following.

Definition 3 (AdaCos). Using the same notation as in Definition 2, let $Y^{(t)} \subset Y$ denote all samples belonging to a mini-batch of size $B \in \mathbb{N}$, i.e. $|Y^{(t)}| = B$. Let $\theta_{x,i} := \arccos(\cos(\phi(x, w), c_i)) \in [0, \pi]$ and the *dynamically adaptive scale parameter* $\hat{s}^{(t)} \in \mathbb{R}_+$ at training step $t \in \mathbb{N}_0$ be set to

$$\hat{s}^{(t)} := \begin{cases} \sqrt{2} \cdot \log(N - 1) & \text{if } t = 0 \\ \frac{\log B_{\text{avg}}^{(t)}}{\cos(\min(\frac{\pi}{4}, \theta_{\text{med}}^{(t)}))} & \text{else} \end{cases} \quad (7)$$

where $\theta_{\text{med}}^{(t)} \in [0, \pi]$ denotes the median of all angles $\theta_{x,i(x)}$ with $x \in X^{(t)}$ and $i(x) \in \{1, \dots, N\}$ such that $l_i(x) = 1$ and

$$B_{\text{avg}}^{(t)} := \frac{1}{B} \sum_{x \in Y^{(t)}} \sum_{\substack{j=1 \\ l_j(x) \neq 1}}^N \exp(\hat{s}^{(t-1)} \cdot \cos(\phi(x, w), c_j)) \quad (8)$$

is the sample-wise average over all summed logits belonging to the non-corresponding classes. Then, the *AdaCos* loss is defined as

$$\begin{aligned} \mathcal{L}_{\text{ada}} : \mathcal{P}(X) \times \mathcal{P}(\mathbb{R}^D) \times \Phi \times W \times L &\rightarrow \mathbb{R}_+ \\ \mathcal{L}_{\text{ada}}(Y, C, \phi, w, l) &:= \mathcal{L}_{\text{ang}}(Y, C, \phi, w, l, \hat{s}, 0). \end{aligned} \quad (9)$$

Remark. When using mixup [58] for data augmentation, $\theta_{\text{med}}^{(t)} \in [0, \pi]$ needs to be replaced with the median of the mixed-up angles as specified in [29].

The AdaCos loss can also be extended to using multiple centers for each class, called sub-clusters, instead of a single one. The idea of using these sub-clusters is to allow the network to learn more complex distributions than a normal distribution for each class enabling the model to have a more differentiated view on the embeddings when using the cosine similarity as an anomaly score. This has been shown to improve the ASD performance [29] and thus helps to differentiate between normal and anomalous samples.

Definition 4 (Sub-cluster AdaCos). Using the same notation as in Definitions 2 and 3, let $C_j \in \mathcal{P}(\mathbb{R}^D)$ with $|C_j| = M$ denote all centers belonging to class $j \in \{1, \dots, N\}$. Let the *dynamically adaptive scale parameter* $\hat{s}^{(t)} \in \mathbb{R}_+$ at training step $t \in \mathbb{N}_0$ be set to

$$\hat{s}^{(t)} := \begin{cases} \sqrt{2} \cdot \log(N \cdot M - 1) & \text{if } t = 0 \\ \frac{f_{\text{max}}^{(t)} + \log \hat{B}_{\text{avg}}^{(t)}}{\cos(\min(\frac{\pi}{4}, \theta_{\text{med}}^{(t)}))} & \text{else} \end{cases} \quad (10)$$

with

$$\hat{B}_{\text{avg}}^{(t)} := \frac{1}{B} \sum_{x \in Y^{(t)}} \sum_{j=1}^N \sum_{c \in C_j} \exp(\hat{s}^{(t-1)} \cos(\phi(x, w), c) - f_{\text{max}}^{(t)}) \quad (11)$$

and

$$f_{\text{max}}^{(t)} := \max_{x \in Y^{(t)}} \max_{j=1}^N \max_{c \in C_j} \hat{s}^{(t-1)} \cdot \cos(\phi(x, w), c). \quad (12)$$

Then, the *sub-cluster AdaCos* loss is defined as

$$\begin{aligned} \mathcal{L}_{\text{sc-ada}} : \mathcal{P}(X) \times \mathcal{P}(\mathcal{P}(\mathbb{R}^D)) \times \Phi \times W \times L &\rightarrow \mathbb{R}_+ \\ \mathcal{L}_{\text{sc-ada}}(Y, C, \phi, w, l) \\ := -\frac{1}{|Y|} \sum_{x \in Y} \sum_{j=1}^N l_j(x) \log(\text{smax}(\hat{s} \cdot \cos(\phi(x, w), C_j))) \end{aligned} \quad (13)$$

where $|C| = N$ and, in this case,

$$\begin{aligned} &\text{smax}(\hat{s} \cdot \cos(\phi(x, w), C_j)) \\ := \sum_{c_j \in C_j} \frac{\exp(\hat{s} \cdot \cos(\phi(x, w), c_j))}{\sum_{k=1}^N \sum_{c_k \in C_k} \exp(\hat{s} \cdot \cos(\phi(x, w), c_k))} \end{aligned} \quad (14)$$

Remark. As shown in [29], for the sub-cluster AdaCos loss as defined above mixup [58] needs to be used. Otherwise, the dynamically adaptive scale parameter $\hat{s}^{(t)}$ grows exponentially.

For the compactness loss, there is no benefit of using sub-clusters. The reason is that an optimal solution of this sub-cluster compactness loss would correspond to the mean of the sub-clusters or, in case all embeddings are normalized, to its projection onto the unit sphere. Hence, there would be a single global optimum and this sub-cluster compactness loss would behave as if only a single sub-cluster is used. For the sub-cluster AdaCos loss, the situation is completely different because the softmax function is applied to all individual sub-clusters and the sum over the resulting scores is taken. This makes the resulting softmax probability, and thus also the loss function, symmetric with respect to the corresponding sub-clusters of an individual class. Therefore, the loss is invariant to changing the position of an embedding on the hypersphere as long as the sum of the distances to the sub-clusters is the same. Hence, also the space of optimal solutions grows with respect to the number of sub-clusters. However, due to the dependence on the sub-clusters of the other classes caused by the softmax function, this invariance is a simplification and the real situation is more complex.

III. RELATION BETWEEN ONE-CLASS LOSSES AND ANGULAR MARGIN LOSSES

For the proof of the main theoretical result of this work, the following basic identity is needed.

Lemma 5. For $x, y \in \mathbb{R}^D$ with $\|x\|_2 = \|y\|_2 = 1$, it holds that

$$\cos(x, y) = 1 - \frac{\|x - y\|_2^2}{2}. \quad (15)$$

Proof. See Appendix.

Remark. This lemma also shows that for normalized embeddings using Euclidean distance and using cosine distance, which in this case is equal to the standard scalar product, are equivalent for computing an anomaly score.

Now, the theorem itself follows.

Theorem 6. Let $Y_j := \{x \in Y : l_j(x) = 1\}$. Then minimizing $\mathcal{L}_{\text{sc-ada}}(Y, C, \phi, w, l)$ with gradient descent minimizes all IC compactness losses with weighted gradients given by

$$\begin{aligned} &\frac{\hat{s}}{2} \sum_{i=1}^N \frac{1}{|Y_i|} \sum_{x \in Y_i} \sum_{c_i \in C_i} P(\tau(\phi(x, w)) = c_i | \tau(\phi(x, w)) \in C_i) \\ &\cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_i\|_2^2 \end{aligned} \quad (16)$$

while maximizing all inter-class compactness losses with weighted gradients given by

$$\begin{aligned} &-\frac{\hat{s}}{2} \sum_{i=1}^N \frac{1}{|Y_i|} \sum_{x \in Y_i} \sum_{k=1}^N \sum_{c_k \in C_k} P(\tau(\phi(x, w)) = c_k) \\ &\cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_k\|_2^2 \end{aligned} \quad (17)$$

where

$$\begin{aligned} &P(\tau(\phi(x, w)) = c_i | \tau(\phi(x, w)) \in C_i) \\ := &\frac{\exp(\hat{s} \cdot \cos(\phi(x, w), c_i))}{\sum_{c'_i \in C_i} \exp(\hat{s} \cdot \cos(\phi(x, w), c'_i))} \end{aligned} \quad (18)$$

and

$$\begin{aligned} &P(\tau(\phi(x, w)) = c_k) \\ := &\frac{\exp(\hat{s} \cdot \cos(\phi(x, w), c_k))}{\sum_{k=1}^N \sum_{c'_k \in C_k} \exp(\hat{s} \cdot \cos(\phi(x, w), c'_k))} \end{aligned} \quad (19)$$

with a cluster assignment function $\tau : \mathbb{R}^D \rightarrow \mathbb{R}^D$ given by

$$\tau(z, C) = \arg \max_{c \in C} \cos(z, c). \quad (20)$$

Proof. Let $x \in Y$, $\phi \in \Phi$ and $\hat{s} \in \mathbb{R}_+$ be fixed and $i \in \{1, \dots, N\}$ such that $l_i(x) = 1$ and $l_j(x) = 0$ for $j \neq i$. To simplify notation, define $e(w, c) := \exp(\hat{s} \cdot \cos(\phi(x, w), c))$. Using Lemma 5, we see that

$$\begin{aligned} &\frac{\partial}{\partial w} \log \left(\sum_{c_i \in C_i} e(w, c_i) \right) \\ = &\frac{\sum_{c_i \in C_i} e(w, c_i) \cdot \hat{s} \cdot \frac{\partial}{\partial w} \cos(\phi(x, w), c_i)}{\sum_{c'_i \in C_i} e(w, c'_i)} \\ = &-\frac{\hat{s}}{2} \sum_{c_i \in C_i} \frac{e(w, c_i) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_i\|_2^2}{\sum_{c'_i \in C_i} e(w, c'_i)} \end{aligned}$$

and similarly

$$\begin{aligned} &\frac{\partial}{\partial w} \log \left(\sum_{k=1}^N \sum_{c_k \in C_k} e(w, c_k) \right) \\ = &\frac{\sum_{k=1}^N \sum_{c_k \in C_k} e(w, c_k) \cdot \hat{s} \cdot \frac{\partial}{\partial w} \cos(\phi(x, w), c_k)}{\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k)} \\ = &-\frac{\hat{s}}{2} \sum_{k=1}^N \sum_{c_k \in C_k} \frac{e(w, c_k) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_k\|_2^2}{\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k)} \\ = &-\frac{\hat{s}}{2} \sum_{c_i \in C_i} \frac{e(w, c_i) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_i\|_2^2}{\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k)} \\ &-\frac{\hat{s}}{2} \sum_{\substack{k=1 \\ k \neq i}}^N \sum_{c_k \in C_k} \frac{e(w, c_k) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_k\|_2^2}{\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k)}. \end{aligned}$$

Using both identities, we obtain

$$\begin{aligned}
& \frac{\partial}{\partial w} \sum_{j=1}^N l_j(x) \log(\text{smax}(\hat{s} \cdot \cos(\phi(x, w), C_j))) \\
&= \frac{\partial}{\partial w} \log \left(\sum_{c_i \in C_i} \frac{e(w, c_i)}{\sum_{k=1}^N \sum_{c_k \in C_k} e(w, c_k)} \right) \\
&= \frac{\partial}{\partial w} \log \left(\sum_{c_i \in C_i} e(w, c_i) \right) - \frac{\partial}{\partial w} \log \left(\sum_{k=1}^N \sum_{c_k \in C_k} e(w, c_k) \right) \\
&= -\frac{\hat{s}}{2} \sum_{c_i \in C_i} \frac{e(w, c_i) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_i\|_2^2}{\sum_{c'_i \in C_i} e(w, c'_i)} \\
&\quad + \frac{\hat{s}}{2} \sum_{c_i \in C_i} \frac{e(w, c_i) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_i\|_2^2}{\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k)} \\
&\quad + \frac{\hat{s}}{2} \sum_{\substack{k=1 \\ k \neq i}}^N \sum_{c_k \in C_k} \frac{e(w, c_k) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_k\|_2^2}{\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k)} \\
&= -\frac{\hat{s}}{2} \left(\sum_{c_i \in C_i} e(w, c_i) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_i\|_2^2 \right. \\
&\quad \cdot \left(\frac{1}{\sum_{c'_i \in C_i} e(w, c'_i)} - \frac{1}{\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k)} \right) \\
&\quad \left. - \sum_{\substack{k=1 \\ k \neq i}}^N \sum_{c_k \in C_k} \frac{e(w, c_k) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_k\|_2^2}{\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k)} \right) \\
&= -\frac{\hat{s}}{2} \left(\sum_{c_i \in C_i} e(w, c_i) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_i\|_2^2 \right. \\
&\quad \cdot \left(\sum_{\substack{k=1 \\ k \neq i}}^N \sum_{c_k \in C_k} \frac{e(w, c_k)}{(\sum_{c'_i \in C_i} e(w, c'_i)) (\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k))} \right) \\
&\quad \left. - \sum_{\substack{k=1 \\ k \neq i}}^N \sum_{c_k \in C_k} \frac{e(w, c_k) \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_k\|_2^2}{\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k)} \right) \\
&= -\frac{\hat{s}}{2} \sum_{\substack{k=1 \\ k \neq i}}^N \sum_{c_k \in C_k} \frac{e(w, c_k)}{\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k)} \\
&\quad \cdot \left(\sum_{c_i \in C_i} \frac{e(w, c_i)}{\sum_{c'_i \in C_i} e(w, c'_i)} \cdot \frac{\partial}{\partial w} \|\phi(x, w) - c_i\|_2^2 \right. \\
&\quad \left. - \frac{\partial}{\partial w} \|\phi(x, w) - c_k\|_2^2 \right) \\
&= -\frac{\hat{s}}{2} \sum_{k=1}^N \sum_{c_k \in C_k} \underbrace{\frac{e(w, c_k)}{\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k)}}_{=P(\tau(\phi(x, w))=c_k)} \\
&\quad \cdot \sum_{c_i \in C_i} \underbrace{\frac{e(w, c_i)}{\sum_{c'_i \in C_i} e(w, c'_i)}}_{=P(\tau(\phi(x, w))=c_i | \tau(\phi(x, w)) \in C_i)} \\
&\quad \cdot \left(\frac{\partial}{\partial w} \|\phi(x, w) - c_i\|_2^2 - \frac{\partial}{\partial w} \|\phi(x, w) - c_k\|_2^2 \right)
\end{aligned}$$

where we used that

$$\begin{aligned}
& \frac{1}{\sum_{c'_i \in C_i} e(w, c'_i)} - \frac{1}{\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k)} \\
&= \frac{\sum_{k=1}^N \sum_{c_k \in C_k} e(w, c_k) - \sum_{c_i \in C_i} e(w, c_i)}{(\sum_{c'_i \in C_i} e(w, c'_i)) (\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k))} \\
&= \sum_{\substack{k=1 \\ k \neq i}}^N \sum_{c_k \in C_k} \frac{e(w, c_k)}{(\sum_{c'_i \in C_i} e(w, c'_i)) (\sum_{k=1}^N \sum_{c'_k \in C_k} e(w, c'_k))}.
\end{aligned}$$

Now, summing over all samples $x \in Y$, normalizing with $|Y|$ and taking the additive inverse yields the desired result.

When using mixup, the right hand side of the last equation needs to be replaced with a weighted sum of two terms, each corresponding to one of the two classes that are mixed-up, because there are $i_1, i_2 \in \{1, \dots, N\}$ such that $l_{i_1}(x) \neq 0 \neq l_{i_2}(x)$. Otherwise, the proof is exactly the same. In conclusion, the proven result still holds for mixed-up samples but includes two similar terms instead of one term. \square

Corollary 7. *Minimizing $\mathcal{L}_{ada}(Y, C, \phi, w, l)$ with gradient descent is equivalent to minimizing*

$$\begin{aligned}
& -\frac{\tilde{s}}{2} \sum_{k=1}^N \text{smax}(\tilde{s} \cdot \cos(\phi(x, w), c_k)) \\
& \cdot \left(\frac{\partial}{\partial w} \|\phi(x, w) - c_i\|_2^2 - \frac{\partial}{\partial w} \|\phi(x, w) - c_k\|_2^2 \right). \tag{21}
\end{aligned}$$

Proof. The proof of Theorem 6 does not depend on the exact structure of the dynamically adaptive scale parameter and thus also holds for the standard AdaCos loss by replacing \hat{s} with \tilde{s} and using only a single sub-cluster for each class. \square

This theorem shows that using an angular margin loss such as the AdaCos loss is essentially the same strategy as proposed in [55] and applied to ASD in [27], i.e. using a compactness loss for increasing IC similarity, as defined in Definition 1, and a so-called descriptiveness loss to decrease inter-class similarity. However, there are differences between both approaches. When minimizing an angular margin loss, inter-class compactness losses are used to decrease inter-class similarity instead of a standard CCE loss. Second, when using two loss functions one usually has to tune a weight parameter to create a weighted sum of both loss terms, which is not needed for an angular margin loss and impossible without access to anomalous samples. Furthermore, the gradients belonging to individual samples are weighted with specific softmax probabilities giving more emphasis the closer the sub-clusters are. As these weights are non-uniform in general, this explicitly shows why using multiple sub-clusters is not equivalent to using a single sub-cluster given by the projection of the mean of the sub-clusters onto the hypersphere as it is the case for an IC compactness loss with multiple sub-clusters. Last but not least, an angular margin loss explicitly ensures a margin between classes, as illustrated in Fig. 1, whereas a combination of compactness losses and a CCE loss only implicitly does this by increasing intra-class similarity. Note that, in [55], inter-class similarity is decreased on another dataset using less relevant classes because only a single class

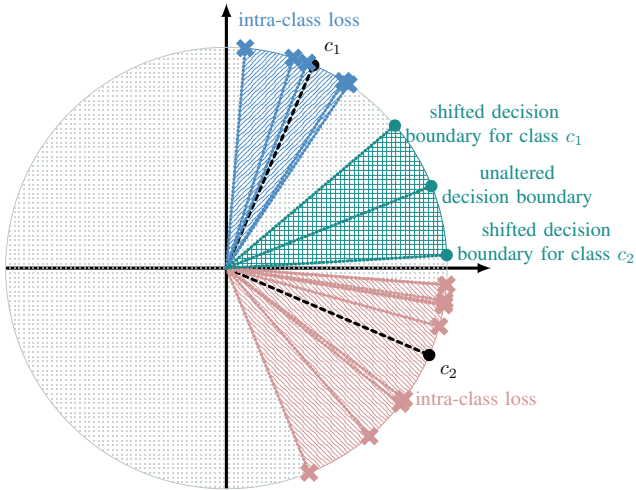


Fig. 1. Illustration of IC compactness losses and the angular margin to be ensured between the classes for $D = 2$, $N = 2$, $M = 1$. Intra-class losses are computed by summing all distances of samples to their corresponding class centers (blue and red areas). Inter-class losses are computed by summing all distances of samples to their corresponding decision boundaries. An unaltered decision boundary is exactly the midpoint between the class centers. When using an angular margin loss, the decision boundaries to the other classes are essentially shifted closer to the class center for which the inter-class loss is computed (see Fig. 1 in [60]). This explicitly ensures a margin between the classes, which is depicted by the green area.

is available on the target dataset. Because of these differences, directly minimizing an angular margin loss leads to a different solution than minimizing a combination of IC losses and a descriptiveness loss.

Note that the IC compactness loss with multiple classes can also be considered a prototypical loss [61] or angular prototypical loss [62] as used for few-shot classification [63], which defines settings where only very few training samples, called shots, are available for each class. The only difference between these prototypical losses and an angular margin loss is that, for prototypical losses, the center vectors are re-calculated as the means of embeddings belonging to corresponding classes by using a so-called support set during training while, for an angular margin loss, the class centers are fixed or adaptable parameters of the network. Hence, this theorem also shows that angular margin losses are a suitable choice for few-shot classification as shown for open-set sound event classification [42] and few-shot keyword spotting [64].

Choosing a classification task as an auxiliary task prevents learning a constant function as a trivial solution. The reason is that, for such a classification task, an optimal solution is a classifier that maps each sample to its corresponding class center and thus corresponds to jointly learning multiple trivial solutions, one for each class, instead of only learning a constant function. As long as each anomalous sample belongs to a well-defined normal class used during training, this optimal solution would yield representations not suitable for detecting anomalies as they would not be distinguishable from representations obtained with normal samples. However, obtaining such a perfect classifier is much more difficult than learning a constant mapping for a single class and thus training

TABLE I
STRUCTURE OF THE DCASE2022 ASD DATASET

subset	split	number of recordings (per section)			
		source domain		target domain	
		normal	anomalous	normal	anomalous
development	training	990	0	10	0
development	test	50	50	50	50
evaluation	training	990	0	10	0
evaluation	test	50	50	50	50

a single model to classify between multiple classes already prevents trivial solutions as long as the classification problem itself is not trivial e.g. by consisting of only a single class. Still, in [33] it has been shown that the ASD performance can be improved by applying the same three strategies as used for the compactness loss [52], namely 1) not using bias terms, 2) not using bounded activation functions and 3) not using trainable class centers. The most likely reason is that these strategies prevent the model to learn trivial solutions, leading to less informative embeddings, for individual classes that are easily recognized.

IV. EXPERIMENTAL RESULTS

Using one-class losses and angular margin losses for ASD will now be compared experimentally.

A. Dataset

For most experiments conducted in this work, the DCASE2022 ASD dataset [4] of the task titled “Unsupervised Anomalous Sound Detection for Machine Condition Monitoring Applying Domain Generalization Techniques” has been used. The dataset consists of recordings of machine sounds with background factory noise. Each recording has a single channel, a length of ten seconds and a sampling rate of 16 kHz and belongs to one of the seven machine types “fan”, “gearbox”, “bearing”, “slide rail”, “valve” from MIMII DG [?] and “toy car”, “toy train” from ToyADMOS2 [65]. For each machine type, there are six different so-called sections each of which is dedicated to a specific type of domain shift. A domain shift means that the characteristics of a machine sound differ in some way between a source domain with many training samples and a target domain with only few training samples. These shifts can be caused by physical changes of the machines e.g. caused by replacing parts for maintenance, or changes in the acoustical environment e.g. a different background noise or using different recording devices. Ideally, the ASD system is able to reliably detect anomalies despite these domain shifts without the need for adapting the system (domain generalization [66]).

The dataset is divided into a development and an evaluation split each containing recordings of 21 sections, three for each machine type. For each recording, information about the machine type and section are given. For the training datasets, domain information (“source” or “target”) and additional attribute information such as states of machine types or noise conditions are given for each recording. For the test datasets, no domain information and no additional attribute information

are given. The exact structure of the dataset can be found in Tab I. The task of an ASD system is to reliably detect anomalous samples regardless of whether a sample belongs to a source or target domain, i.e. using a single decision threshold for both domains of a section.

Some of the experiments have also been conducted on the DCASE2023 ASD dataset [67], [68] belonging to the task “First-Shot Unsupervised Anomalous Sound Detection for Machine Condition Monitoring”. Similar to the DCASE2022 ASD dataset, this dataset is also aimed at domain generalization for ASD with the following differences. First and foremost, the development and evaluation split of the dataset contain different machine types. The development set contains the same machine types as the DCASE2022 dataset, namely “fan”, “gearbox”, “bearing”, “slide rail”, “valve” from MIMII DG [?] and “toy car”, “toy train” from ToyADMOS2 [65]. The evaluation set contains seven completely different machine types, namely “toy drone”, “n-scale toy train”, “vacuum”, and “toy tank” from [69] and “bandsaw”, “grinder”, “shaker” from [?]. Furthermore, for each machine type there is only a single section. This lowers the difficulty of the auxiliary classification task and thus makes it more difficult to extract embeddings, which are sensitive to anomalous changes of the target sounds.

For the DCASE ASD datasets, two performance measures are used to evaluate the performance of individual ASD systems. One metric is the area under the receiver operating characteristic (ROC) curve (AUC), the other metric is the partial area under the ROC curve (pAUC) [70], which is the AUC calculated over a low false positive rate ranging from 0 to p with $p = 0.1$ in this case. The pAUC is used as an additional metric because decision thresholds for machine condition monitoring are usually set to a value that gives a low number of false alarms and thus this area of the ROC curve is of particular interest. Both are threshold-independent metrics allowing a more objective comparison between different ASD systems than threshold-dependent metrics [1], [71].

B. System Description

The focus of this work is to explain why angular margin losses work well for ASD. This requires using different loss functions for training an ASD system. To this end, the conceptually simple state-of-the-art system presented in [33], which only consists of a single model and uses the same settings for all machine types, is utilized. For all experiments conducted in this work, only the loss function used for training the system is altered. The system utilizes a magnitude spectrogram as well as the whole magnitude spectrum as input representations and uses two different convolutional sub-models for handling these, resulting in two different embeddings. Then, both embeddings are concatenated to obtain a single embedding and the sub-cluster AdaCos loss [29] is applied with 16 sub-clusters, which are initialized uniformly at random, for training the model. For the magnitude spectrogram, temporal mean normalization is applied to reduce the effect of different acoustic domains and make both input feature representations a bit more different by removing constant frequency information from the spectrograms. Furthermore, the model does not use

bias terms or trainable clusters as this improves the ASD performance by avoiding trivial solutions as discussed before. The model is trained for 10 epochs with a batch size of 64 using mixup [58] with a uniform distribution for sampling the mixing coefficient and is implemented in Tensorflow [72].

After training the model using an auxiliary classification task, embeddings are extracted for the recordings. For each section of the dataset, k-means with $k = 16$ is applied to all normal training samples belonging to the source domain of this section. The goal is to represent the distribution of the normal embeddings and be able to compute an anomaly score by taking the minimum cosine distance to the mean embeddings belonging to the same section as a given test sample. Note that these means do not correspond to the sub-clusters as some sub-clusters may not have been used by the network during training. It is possible that the embeddings are clustered between the sub-clusters due to the complex dependence between the sub-clusters of the other classes. Still, it has been shown taking the same number of clusters usually performs best [29]. Since there are only 10 normal samples available for the target domain, the minimum over the direct cosine distances to the corresponding embeddings is used. As a last step, the minimum of the minimum cosine distances belonging to both domains is used to have an ASD system that generalizes to both domains. Hence, a higher anomaly score indicates anomalous sounds whereas a smaller value indicates normal sounds. More details about the system including a hyperlink to an open-source implementation can be found in [33].

C. Performance Evaluations

Regardless of the loss function, training the ASD model without using anomalous samples is not directly targeting the ASD performance but only indirectly since the auxiliary task is aimed at obtaining embeddings suitable for ASD. Although, there is a strong relation between the auxiliary and the ASD task, as otherwise training an ASD model by using an auxiliary task would not lead to usable representations, the actual ASD performance needs to be evaluated experimentally and cannot be investigated theoretically because there are no anomalous samples available during training. Therefore, the resulting ASD performances obtained by minimizing both types of loss functions, angular margin losses and one-class losses, using individual auxiliary classification tasks will be evaluated first. Furthermore, a combined loss consisting of the sum of the mean of the IC compactness losses and an additional softmax layer with a CCE loss for classification, as proposed in [55], is evaluated. The results can be found in Tab. II. Note that it is also possible to divide the classification task into several different classification tasks as for example one task for the machine type and other ones for all or specific attributes [30], [31]. However, in our experience this does not improve performance unless weights for the losses belonging to different machine types are manually tuned to improve the ASD performance. Since this requires access to anomalous samples, tuning these weights is impossible in a truly semi-supervised setting.

TABLE II

ASD PERFORMANCE OBTAINED WITH DIFFERENT LOSSES USING DIFFERENT AUXILIARY TASKS. HARMONIC MEANS OF ALL AUCs AND pAUCs OVER ALL PRE-DEFINED SECTIONS OF THE DATASET ARE DEPICTED IN PERCENT. ARITHMETIC MEAN AND STANDARD DEVIATION OF THE RESULTS OVER FIVE INDEPENDENT TRIALS ARE SHOWN. BEST RESULTS IN EACH COLUMN ARE HIGHLIGHTED WITH BOLD LETTERS.

DCASE2022 development set							
loss	classes of auxiliary task (number of classes)	source domain		target domain		both domains	
		AUC	pAUC	AUC	pAUC	AUC	pAUC
IC compactness loss (Def. 1)	none (1)	56.4 ± 1.4	53.9 ± 0.6	53.6 ± 0.9	52.6 ± 0.3	55.1 ± 1.2	52.6 ± 0.4
IC compactness loss (Def. 1)	machine types (7)	66.5 ± 2.9	60.6 ± 0.6	63.6 ± 2.2	57.1 ± 0.9	65.0 ± 1.7	57.8 ± 0.6
IC compactness loss (Def. 1)	machine types and sections (42)	77.6 ± 1.7	70.5 ± 0.9	75.3 ± 0.9	63.3 ± 0.8	76.4 ± 0.9	63.5 ± 0.6
IC compactness loss (Def. 1)	machine types and sections, models trained individually (1)	50.0 ± 2.3	52.1 ± 0.6	51.7 ± 1.8	52.2 ± 0.4	51.8 ± 1.8	51.4 ± 0.4
IC compactness loss (Def. 1)	machine types, sections and attribute information (342)	80.7 ± 1.9	73.7 ± 1.0	74.5 ± 0.9	62.1 ± 1.2	78.1 ± 0.8	63.3 ± 0.9
IC compactness loss (Def. 1) + CCE	machine types, sections and attribute information (342)	82.5 ± 0.7	75.2 ± 0.7	75.5 ± 0.6	61.2 ± 1.6	79.0 ± 0.6	64.8 ± 0.9
AdaCos loss (Def. 3)	machine types, sections and attribute information (342)	83.0 ± 1.3	75.2 ± 1.8	75.4 ± 1.0	60.9 ± 0.8	79.2 ± 0.9	64.3 ± 0.7
sub-cluster AdaCos loss (Def. 4)	machine types, sections and attribute information (342)	84.2 ± 0.8	76.5 ± 0.9	78.5 ± 0.9	62.5 ± 0.9	81.4 ± 0.7	66.6 ± 0.9
DCASE2022 evaluation set							
loss	classes of auxiliary task (number of classes)	source domain		target domain		both domains	
		AUC	pAUC	AUC	pAUC	AUC	pAUC
IC compactness loss (Def. 1)	none (1)	49.9 ± 0.8	50.6 ± 0.4	51.0 ± 0.4	51.0 ± 0.7	50.9 ± 0.5	50.3 ± 0.4
IC compactness loss (Def. 1)	machine types (7)	59.6 ± 1.3	56.9 ± 0.5	57.6 ± 1.8	53.8 ± 0.9	59.3 ± 1.5	54.6 ± 0.6
IC compactness loss (Def. 1)	machine types and sections (42)	70.8 ± 1.2	62.1 ± 0.7	61.7 ± 0.8	55.4 ± 1.0	66.3 ± 0.6	56.5 ± 0.4
IC compactness loss (Def. 1)	machine types and sections, models trained individually (1)	52.9 ± 1.4	51.7 ± 0.5	54.5 ± 0.6	51.6 ± 0.3	54.2 ± 0.8	51.2 ± 0.3
IC compactness loss (Def. 1)	machine types, sections and attribute information (342)	73.7 ± 0.5	63.4 ± 0.7	67.9 ± 1.0	57.8 ± 1.3	70.9 ± 0.6	58.5 ± 0.9
IC compactness loss (Def. 1) + CCE	machine types, sections and attribute information (342)	74.7 ± 0.7	64.9 ± 1.1	69.2 ± 0.7	59.8 ± 1.3	71.9 ± 0.6	59.5 ± 1.0
AdaCos loss (Def. 3)	machine types, sections and attribute information (342)	76.3 ± 1.0	66.0 ± 0.5	69.9 ± 0.8	59.9 ± 1.5	73.2 ± 0.4	60.1 ± 0.9
sub-cluster AdaCos loss (Def. 4)	machine types, sections and attribute information (342)	76.8 ± 0.8	65.8 ± 0.2	69.8 ± 0.5	59.7 ± 1.1	73.4 ± 0.5	59.8 ± 0.8
DCASE2023 development set							
loss	classes of auxiliary task (number of classes)	source domain		target domain		both domains	
		AUC	pAUC	AUC	pAUC	AUC	pAUC
IC compactness loss (Def. 1)	none (1)	50.7 ± 3.5	52.6 ± 0.3	45.3 ± 1.9	50.1 ± 0.5	48.9 ± 1.4	50.9 ± 0.4
IC compactness loss (Def. 1)	machine types (14)	67.3 ± 2.7	63.0 ± 1.4	67.8 ± 1.2	58.6 ± 1.1	67.4 ± 1.4	59.4 ± 1.1
IC compactness loss (Def. 1)	machine types, models trained individually (1)	46.7 ± 1.9	51.7 ± 0.6	45.9 ± 3.2	50.4 ± 0.8	47.6 ± 2.1	50.7 ± 0.6
IC compactness loss (Def. 1)	machine types and attribute information (186)	67.6 ± 2.5	61.6 ± 1.2	70.0 ± 2.4	56.4 ± 1.9	68.3 ± 1.9	57.1 ± 1.3
IC compactness loss (Def. 1) + CCE	machine types and attribute information (186)	70.1 ± 1.5	63.3 ± 1.3	71.0 ± 1.3	55.5 ± 1.1	70.4 ± 1.0	56.7 ± 0.8
AdaCos loss (Def. 3)	machine types and attribute information (186)	69.8 ± 1.5	62.8 ± 1.3	72.1 ± 1.2	55.4 ± 1.7	71.2 ± 0.7	56.8 ± 1.2
sub-cluster AdaCos loss (Def. 4)	machine types and attribute information (186)	69.4 ± 1.5	61.4 ± 1.5	72.4 ± 1.6	55.3 ± 1.2	71.0 ± 1.2	56.3 ± 1.1
DCASE2023 evaluation set							
loss	classes of auxiliary task (number of classes)	source domain		target domain		both domains	
		AUC	pAUC	AUC	pAUC	AUC	pAUC
IC compactness loss (Def. 1)	none (1)	51.8 ± 2.1	51.4 ± 1.2	50.0 ± 1.9	50.5 ± 0.7	51.6 ± 0.9	50.8 ± 0.6
IC compactness loss (Def. 1)	machine types (14)	59.3 ± 1.9	54.4 ± 0.6	54.3 ± 2.1	51.2 ± 0.5	56.7 ± 1.2	52.0 ± 0.6
IC compactness loss (Def. 1)	machine types, models trained individually (1)	51.3 ± 0.7	51.9 ± 0.7	54.7 ± 1.7	52.3 ± 0.8	53.2 ± 1.0	51.5 ± 0.6
IC compactness loss (Def. 1)	machine types and attribute information (186)	73.0 ± 1.9	62.1 ± 1.4	58.9 ± 2.7	55.1 ± 1.4	64.1 ± 1.8	55.6 ± 0.8
IC compactness loss (Def. 1) + CCE	machine types and attribute information (186)	62.6 ± 1.4	62.5 ± 1.9	62.2 ± 2.6	56.2 ± 0.8	67.2 ± 0.7	58.0 ± 0.9
AdaCos loss (Def. 3)	machine types and attribute information (186)	72.3 ± 1.7	62.1 ± 1.4	61.6 ± 3.1	56.4 ± 1.1	67.0 ± 1.5	57.4 ± 0.9
sub-cluster AdaCos loss (Def. 4)	machine types and attribute information (186)	72.1 ± 1.9	61.3 ± 1.5	62.3 ± 2.7	56.0 ± 0.7	67.3 ± 1.3	57.4 ± 0.7

It can be seen that for both datasets the ASD performance improves with the number of classes being used for the auxiliary task. When using only a single class for all data or for individual machine types and sections, the AUC is close to 50%, which corresponds to randomly guessing whether a sample is anomalous or not. The most likely reason for this is the factory background noise contained in the recordings, which is highly diverse and contains many sound sources other than the target machine. A model trained with a one-class loss does not know the difference between the sound events emitted by the machines to be monitored and any other sounds contained in the recordings. The more complex (in terms of numbers of classes) the chosen auxiliary task is, the more information needs to be captured inside the embeddings for solving this task. Additionally, the background noise does not contain any helpful information for learning to discriminate between the classes defined by the auxiliary task assuming the noise is not class-specific. As a result, the model learns to monitor specific frequencies or temporal patterns important for specific machine types with specific settings and thus also learns to ignore the background noise and to isolate sounds emitted by the targeted machines. Furthermore, it can be

observed that using an explicit classification task improves performance on all dataset splits. Ensuring an angular margin between the classes slightly improves the overall performance, but not significantly, often leading to very similar results. The most likely reason is that by increasing intra-class similarity implicitly introduces a margin between different classes. Still, using an angular margin loss does not have any drawbacks over using a compactness and a descriptiveness loss. As a last observation, the sub-cluster AdaCos loss performs slightly better than the AdaCos loss on the development split of the DCASE2022 dataset while yielding a similar performance on the other dataset splits. A possible explanation that there are no significant improvements on the DCASE2023 datasets when using an angular margin loss is that the auxiliary classification task is not as difficult as for the DCASE2022 dataset because there is only one section for each machine type. Slight improvements in performance when using multiple sub-clusters for the AdaCos loss have been observed on the DCASE2020 dataset [2] in [29]. Note that the DCASE2020 dataset only contains machine recordings with a single parameter setting for each section and no domain shifts, i.e. consists of a single source domain, and thus the task is very different from the

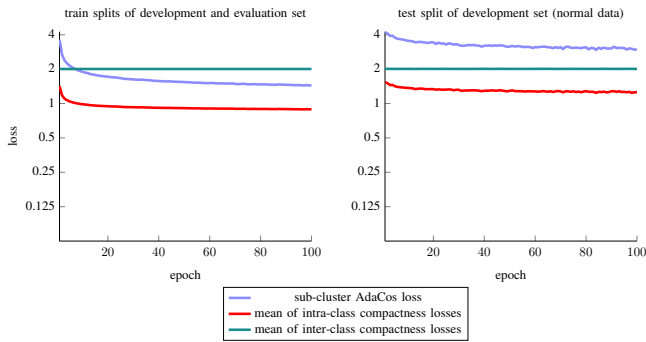


Fig. 2. Different losses after each epoch when training by minimizing sub-cluster AdaCos with a single sub-cluster per class and using mixup.

much more difficult task considered here. In conclusion, an angular margin loss for ASD in combination with an auxiliary classification task that uses as many meaningful classes as possible is an excellent choice when training an ASD system based on audio embeddings.

In the previous paragraph, we made the assumption that the noise is not class-specific. However, if there is a single class with very specific noise that is only present for this particular class or, even worse, if this is the case for all classes, then an auxiliary classification task will very likely not improve the results. The reason is that the model does not learn to closely monitor the machine sound because also the background noise contains useful information for discriminating between the classes. Therefore, assuming that the noise is not class-specific is essential and intuitively makes sense for machine condition monitoring as one would expect that at least some machines share the same noise distribution when running in the same factory or acoustic environment. Moreover, as shown in Theorem 6, minimizing an angular margin loss using an auxiliary classification task also explicitly increases intra-class similarity. Hence, even if the noise is class-specific and thus the auxiliary classification task does not aid the ASD task, the performance is still at least as good as when not using a classification task at all but only minimizing the intra-class compactness losses and there should not be a disadvantage.

D. Minimizing Compactness Loss by Minimizing an Angular Margin Loss

In Theorem 6, it has been shown that minimizing an angular margin loss also minimizes all IC compactness losses and maximizes all inter-class compactness losses. This fact is now verified experimentally by training a model using the sub-cluster AdaCos loss while also monitoring all compactness losses. The results are depicted in Fig. 2 and Fig. 3. Regardless of the dataset splits and regardless of using or not using mixup, the angular margin loss and the mean of the IC compactness losses are decreasing during training. The mean of the inter-class compactness loss is constantly equal to 2, even without training. The reason is that all sub-cluster centers in this work are constant, randomly initialized and projected to the unit sphere. Hence, By Lemma 5, a squared Euclidean distance of 2 corresponds to an angle of $\frac{\pi}{2}$, i.e. orthogonality. The most

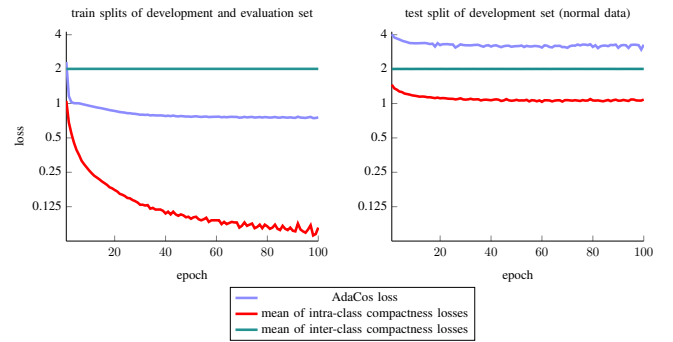


Fig. 3. Different losses after each epoch when training by minimizing AdaCos and not using mixup.

likely reason is that the randomly initialized center vectors are approximately orthogonal with very high probability because of the high dimension $D = 256$ of the embedding space. Thus, samples that are similar to the center of one class will be approximately orthogonal to the centers of the other classes. Overall, this is exactly the expected behavior as predicted by Theorem 6 and therefore verifies the theoretical results. Note that smaller loss values do not correspond to a better ASD performance because minimizing these losses only optimizes the performance for the auxiliary task, which is not the same as the ASD task.

E. Visualizing Normal and Anomalous Regions in Input Representations as Perceived by the System

To further investigate the effect of using an auxiliary task with multiple classes, another experiment using RISE [73] is carried out. RISE highlights regions of the input representations that are considered normal or anomalous by the ASD system. Our goal is to show that utilizing an auxiliary classification task for training the system, as done when minimizing an angular margin loss, enables the system to closely monitor specific machine sounds by focusing on regions belonging to specific patterns of the input data. Although the ASD performance is worse when only using spectrograms as input representations [33], for these experiments a model using only spectrograms as input has been trained. The reason is that these representations are visually more appealing for the human eye than waveforms or spectra and thus more suitable to visually highlight normal and anomalous regions.

To visualize areas of the input representation responsible for a decision, RISE masks random entries of the spectrograms using binary masks and evaluates the ASD score using the masked spectrogram. This step is repeated for many iterations. Then, the sum of the masks weighted with the corresponding ASD scores is taken and normalized with the expected value of a random binary mask, which depends on the chosen sampling distribution. The result is called an *importance map* and visualizes the impact of specific regions of a spectrogram on the resulting anomaly score.

The problem is that the dimension of the spectrograms is very high because a time dimension of $T = 311$ and a frequency dimension of $F = 513$ is used. Thus, there are

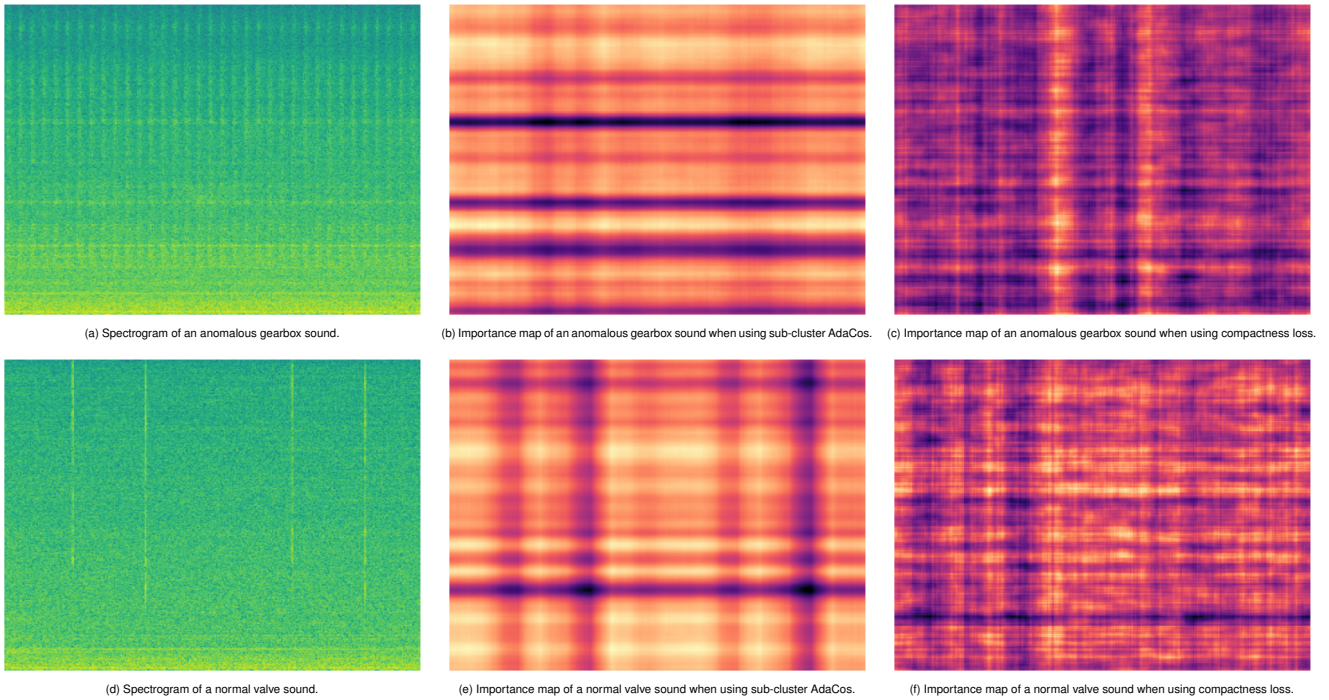


Fig. 4. Log scaled spectrograms (left column), importance maps obtained with RISE when training with the sub-cluster AdaCos loss and classifying between different machine types, sections and attribute information (middle column), and importance maps obtained with RISE when training with an IC compactness loss and no auxiliary classification task (right column) for two different recordings belonging to the test split of the development set (rows). For the importance maps, blue colors indicate normal regions and yellow colors indicate regions that are found to be anomalous by the model. All subfigures use individual color scales to improve visual appearance for differently scaled importance maps and thus colors of different subfigures cannot be compared to each other.

$2^{T \cdot F} = 2^{159543}$ possible binary masks and thus RISE requires clearly too many iterations. To significantly reduce the search space from $2^{F \cdot T}$ to 2^{F+T} , individual time and frequency masks are randomly generated with a probability of 0.25 for a time step or frequency bin to be masked and both masks are combined by element-wise multiplication. This restriction is not too severe because most sounds emitted by machines are relatively stable over time with specific frequencies (e.g. fans), consist of multiple stable sound events with on- and offsets (e.g. slide rails) or only consist of short sound events over a wide frequency range with a specific temporal structure (e.g. valves). For further reduction of the search space, small binary masks are generated and then up-sampled and randomly cropped to match the dimension of the spectrogram to be masked as proposed in [73]. More concretely, we used time masks of size 20 and frequency masks of size 34 resulting in a search space of 2^{54} , which is still very large but much smaller than before. For generating a single importance map, 640,000 iterations have been used.

Magnitude spectrograms (visualized in log scale) and corresponding importance maps belonging to two different samples using i) a model trained with an IC compactness loss without an auxiliary task, and ii) a model trained with the sub-cluster AdaCos loss and an auxiliary task for classifying between different machine types, sections and attribute information are depicted in Fig. 4. For the depicted importance maps, blue colors indicate normal regions and yellow colors indicate anomalous regions as perceived by the system. Note that, since the system does not yield perfect results, these regions do not

need to really belong to normal and anomalous regions. As there are only binary labels, indicating normal or anomalous samples, available for each entire audio recording and we are no subject matter experts for machine condition monitoring, we do not know which regions are normal or anomalous. Still, for the purpose of showing that utilizing meta information when training a model, as done by angular margin losses, helps the system to have a better understanding of the structure of the data these plots are sufficient. There are several observations to be made. Comparing the representations depicted in Fig. 4b and 4e with the ones depicted in Fig. 4c and 4f, we suggest that using sub-cluster AdaCos, i.e. Fig. 4b and 4e, more clearly shows time and frequency structures at a resolution correlating with the structures resp. acoustic events visible in the spectrograms depicted in Fig. 4a and 4d.

For the anomalous gearbox example (Fig. 4a), the importance map depicted in Fig. 4b shows that specific frequencies are monitored and considered to be normal or anomalous. Interestingly, the normal frequency regions (in blue) in Fig. 4b exactly correspond to the frequencies containing high energy (Fig. 4a) showing that the model expects a gearbox sound from this section to have high energy in these regions. The frequencies that are considered most anomalous, which mostly corresponds to the frequency range between the bottom two normal frequency bands, only contain some energy. This indicates that a normal machine sound should either contain no energy or much more energy for these frequencies. In contrast to this, the importance map depicted in Fig. 4c does not monitor specific frequencies and the only clearly visible

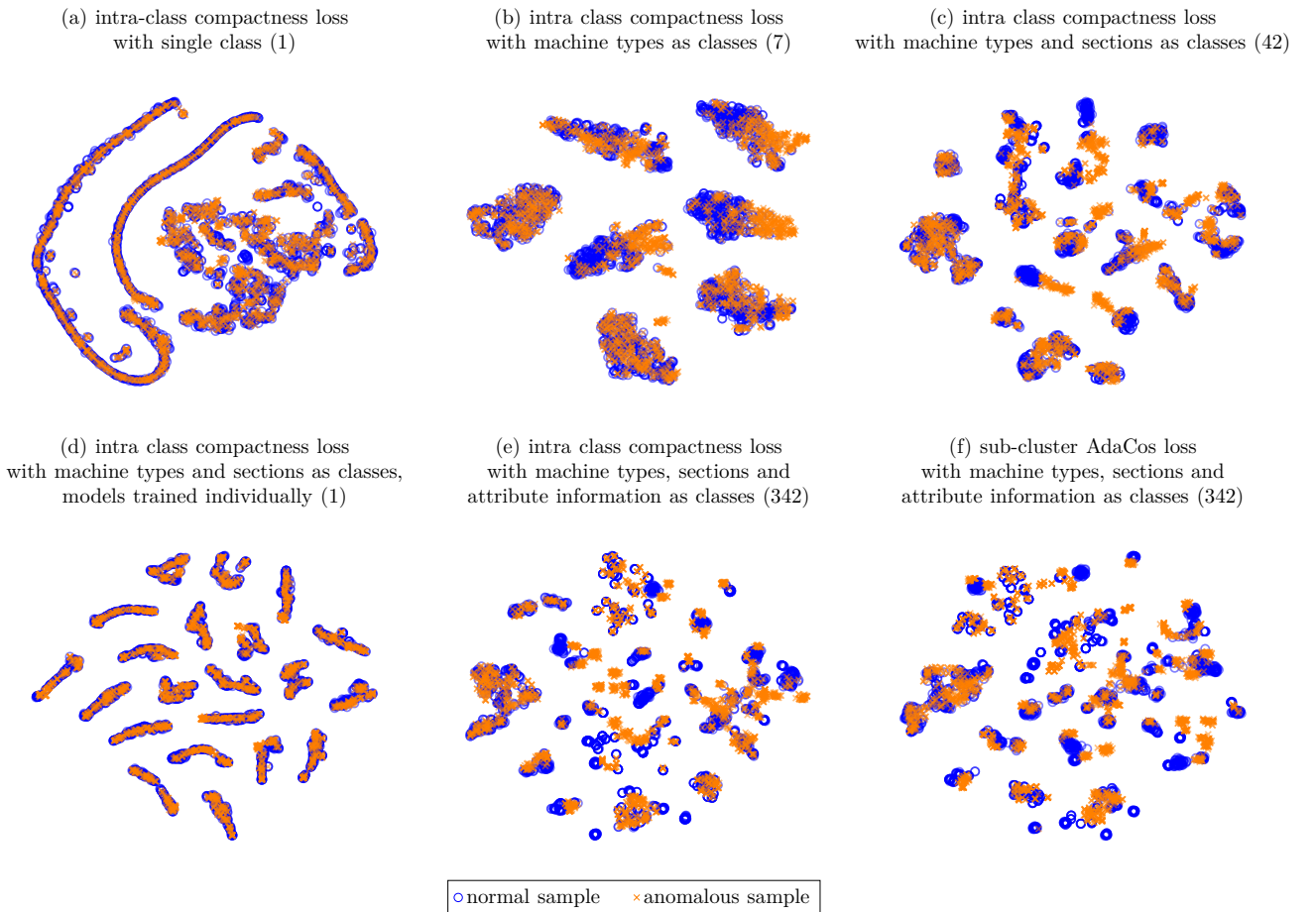


Fig. 5. Visualizations of the test split of the development set in the learned embedding space for different loss functions and auxiliary tasks using t-SNE. Numbers in brackets denote the number of different classes used for the auxiliary task.

structures are two vertical lines indicating anomalous regions (in yellow). Although we cannot guarantee that the regions in the spectrogram corresponding to these vertical lines are not anomalous, at least visually there is no energy present in these locations. Since the recordings of the machine sounds do not start and end at the same fixed time steps, it does not make sense that the model expects temporal patterns at exactly these time steps that are missing and to thus consider such patterns to be anomalous. Therefore, it seems that these structures are errors of the model.

The importance maps belonging to the normal valve example (Fig. 4d) show a similar behavior but for temporal patterns in addition to specific frequencies. Here, the main four normal vertical patterns in the importance map shown in Fig. 4e correspond the four high energy patterns of the spectrogram showing that the system views these temporal patterns as normal for a valve sound. In contrast, the importance map depicted in Fig. 4f does not show that the system has learned to detect these patterns and looks almost random.

Overall, the depicted results add further confidence to the claim that training a model with an auxiliary classification task with many classes enables the model to learn much more meaningful embeddings, also leading to much better capabilities for detecting anomalous sound events than a model

TABLE III
MEAN AND STANDARD DEVIATION OF THE AVERAGE EUCLIDEAN DISTANCE BETWEEN THE T-SNE PROJECTIONS OF EACH ANOMALOUS SAMPLE AND THE CLOSEST NORMAL SAMPLE OVER FIVE TRIALS FOR DIFFERENT LOSSES AND USING DIFFERENT AUXILIARY TASKS.

loss	classes of auxiliary task (number of classes)	average distance
IC compactness loss	none (1)	0.485 ± 0.007
IC compactness loss	machine types (7)	1.636 ± 0.037
IC compactness loss	machine types and sections (42)	2.175 ± 0.075
IC compactness loss	machine types and sections, models trained individually (1)	0.559 ± 0.002
IC compactness loss	machine types, sections and attribute information (342)	2.646 ± 0.045
sub-cluster AdaCos loss	machine types, sections and attribute information (342)	2.947 ± 0.022

trained with only a single class.

F. Visualizing the Resulting Embedding Spaces Using t-SNE

As a last experiment, the embedding spaces resulting from using different loss functions and auxiliary tasks are visualized in Figure 5 using t-SNE [74]. Note that by Lemma 5 it does not matter whether t-SNE is evaluated with the cosine distance or the Euclidean distance because both are equivalent when determining the degree of similarity between samples on the unit sphere. It can be seen that using more classes for the auxiliary task helps to separate normal and anomalous samples (Fig. 5b,c,e,f). When only using a single class (Fig. 5a) or individually trained models (Fig. 5d), there is no visual difference

between normal and anomalous samples. However, it can also be seen that the model has not learned a trivial solution as the embedding spaces did not collapse to a single fixed point, which would correspond to a uniformly distributed t-SNE embedding space. Moreover, the ASD performance would be very close to 50% as normal and anomalous samples would be indistinguishable in the embedding space. Therefore, the applied regularization strategies, namely not using trainable centers and not using bias terms, work and a completely failed regularization is not the main underlying problem. These visual impressions are verified by computing the average Euclidean distance between each anomalous sample and the closest normal sample in the t-SNE embedding space. The results can be found in Tab. III and also agree with the performance results shown in Table II. Note that the distance in the original embedding space is implicitly captured by the ASD performance given in II because the anomaly score is computed by taking the distance to the closest normal sample in the target domain and the closest mean in the source domain. Again, the most likely explanation for the strong differences between the embedding spaces in terms of ASD capabilities is that using multiple classes enables the model to focus less on or even ignore the background noise and isolate the targeted machine sounds. This helps the model to more robustly detect deviations from normal machine sounds despite the acoustically noisy recording conditions and thus results in better ASD performance.

V. CONCLUSIONS

In this work, it has been investigated why using angular margin losses works well for semi-supervised ASD. To this end, it has been shown, both theoretically and experimentally, that reducing an angular margin loss also minimizes the IC compactness loss while simultaneously maximizing the inter-class compactness loss. Therefore, angular margin losses in combination with an auxiliary classification task can be viewed as regularized one-class losses preventing the model to learn trivial solutions. In experiments conducted on the DCASE2022 and DCASE2023 ASD datasets for machine condition monitoring, it has been shown that using an auxiliary task with as many meaningful classes as possible and using an angular margin loss leads to significantly better ASD performance than using a one-class loss such as the IC compactness loss. Furthermore, RISE has been applied to create importance maps for different losses and t-SNE has been used to visualize the resulting embedding spaces. All the conducted experiments show that by using an angular margin the model used for extracting the embeddings learns to monitor relevant frequency bins and learns machine-specific temporal patterns. This enables the model to isolate machine sounds and effectively ignore background noise present in the recording explaining why angular margin losses with an auxiliary task are a good choice for training an ASD system.

For future work, it is planned to investigate whether using auxiliary tasks based on self-supervised learning to obtain suitable representations of the data improves the resulting ASD performance. In addition, sophisticated methods for

visualizing anomalous regions of input representations should be developed as being able to localize these regions is very useful for practical applications and theoretical analysis of ASD systems.

ACKNOWLEDGMENTS

We would like to thank Paul M. Baggenstoss and Lukas Henneke as well as the anonymous reviewers for their valuable comments that improved the quality of this work.

APPENDIX

PROOF OF LEMMA 5

Using only basic definitions, we obtain

$$\begin{aligned} \|x - y\|_2^2 &= \sum_{i=1}^D (x_i - y_i)^2 \\ &= \sum_{i=1}^D x_i^2 + \sum_{i=1}^D y_i^2 - 2 \sum_{i=1}^D x_i y_i \\ &= \|x\|_2^2 + \|y\|_2^2 - 2\langle x, y \rangle \\ &= 2 \left(1 - \frac{\langle x, y \rangle}{\|x\|_2 \|y\|_2} \right) \\ &= 2(1 - \cos(\angle(x, y))), \end{aligned}$$

which finishes the proof. \square

REFERENCES

- [1] C. Aggarwal, *Outlier Analysis*, 2nd ed. Springer, 2017.
- [2] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, and N. Harada, "Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 81–85.
- [3] Y. Kawaguchi, K. Imoto, Y. Koizumi, N. Harada, D. Niizumi, K. Dohi, R. Tanabe, H. Purohit, and T. Endo, "Description and discussion on DCASE 2021 challenge task 2: Unsupervised anomalous detection for machine condition monitoring under domain shifted conditions," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2021, pp. 186–190.
- [4] K. Dohi, K. Imoto, N. Harada, D. Niizumi, Y. Koizumi, T. Nishida, H. Purohit, R. Tanabe, T. Endo, M. Yamamoto, and Y. Kawaguchi, "Description and discussion on DCASE 2022 challenge task 2: Unsupervised anomalous sound detection for machine condition monitoring applying domain generalization techniques," in *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022, pp. 26–30.
- [5] S. N. Murthy and E. Agu, "Deep learning anomaly detection methods to passively detect COVID-19 from audio," in *International Conference on Digital Health (ICDH)*. IEEE, 2021, pp. 114–121.
- [6] T. Dissanayake, T. Fernando, S. Denman, S. Sridharan, H. Ghaemmaghami, and C. Fookes, "A robust interpretable deep learning classifier for heart anomaly detection without segmentation," *IEEE J. Biomed. Health Informatics*, vol. 25, no. 6, pp. 2162–2171, 2021.
- [7] S. Ntalampiras and I. Potamitis, "Acoustic detection of unknown bird species and individuals," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 3, pp. 291–300, 2021.
- [8] T. Cejrowski and J. Szymanski, "Buzz-based honeybee colony fingerprint," *Comput. Electron. Agric.*, vol. 191, p. 106489, 2021.
- [9] C. Zieger, A. Brutti, and P. Svaizer, "Acoustic based surveillance system for intrusion detection," in *Sixth International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2009, pp. 314–319.
- [10] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, "Audio surveillance of roads: A system for detecting anomalous sounds," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 1, pp. 279–288, 2016.

- [11] Y. Li, X. Li, Y. Zhang, M. Liu, and W. Wang, "Anomalous sound detection using deep audio representation and a BLSTM network for audio surveillance of roads," *IEEE Access*, vol. 6, pp. 58 043–58 055, 2018.
- [12] G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci, and A. Sarti, "Scream and gunshot detection and localization for audio-surveillance systems," in *Fourth International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2007, pp. 21–26.
- [13] T. Hayashi, T. Komatsu, R. Kondo, T. Toda, and K. Takeda, "Anomalous sound event detection based on wavenet," in *26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 2494–2498.
- [14] S. Shon, N. Dehak, D. A. Reynolds, and J. R. Glass, "MCE 2018: The 1st multi-target speaker detection and identification challenge evaluation," in *20th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2019, pp. 356–360.
- [15] A. Mesaros, T. Heittola, and T. Virtanen, "Acoustic scene classification in DCASE 2019 challenge: Closed and open set classification and data mismatch setups," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2019, pp. 164–168.
- [16] J. Naranjo-Alcazar, S. Perez-Castanos, P. Zuccarello, A. M. Torres, J. J. Lopez, F. J. Ferri, and M. Cobos, "An open-set recognition and few-shot learning dataset for audio event classification in domestic environments," *Pattern Recognition Letters*, 2022.
- [17] E. Marchi, F. Vesperini, S. Squartini, and B. W. Schuller, "Deep recurrent neural network-based autoencoders for acoustic novelty detection," *Comput. Intell. Neurosci.*, vol. 2017, pp. 4 694 860:1–4 694 860:14, 2017.
- [18] Y. Koizumi, S. Saito, H. Uematsu, Y. Kawachi, and N. Harada, "Unsupervised detection of anomalous sound based on deep learning and the neyman-pearson lemma," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 27, no. 1, pp. 212–224, 2019.
- [19] K. Suefusa, T. Nishida, H. Purohit, R. Tanabe, T. Endo, and Y. Kawaguchi, "Anomalous sound detection based on interpolation deep neural network," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 271–275.
- [20] R. Giri, F. Cheng, K. Helwani, S. V. Tenneti, U. Isik, and A. Krishnaswamy, "Group masked autoencoder based density estimator for audio anomaly detection," in *5th the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 51–55.
- [21] S. Kapka, "Id-conditioned auto-encoder for unsupervised anomaly detection," in *5th the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 71–75.
- [22] G. Wichern, A. Chakrabarty, Z. Wang, and J. L. Roux, "Anomalous sound detection using attentive neural processes," in *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2021, pp. 186–190.
- [23] K. Dohi, T. Endo, H. Purohit, R. Tanabe, and Y. Kawaguchi, "Flow-based self-supervised density estimation for anomalous sound detection," in *International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2021, pp. 336–340.
- [24] K. Dohi, T. Endo, and Y. Kawaguchi, "Disentangling physical parameters for anomalous sound detection under domain shifts," in *30th European Signal Processing Conference EUSIPCO*. IEEE, 2022, pp. 279–283.
- [25] R. Giri, S. V. Tenneti, F. Cheng, K. Helwani, U. Isik, and A. Krishnaswamy, "Self-supervised classification for detecting anomalous sounds," in *5th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 46–50.
- [26] J. A. Lopez, H. Lu, P. Lopez-Meyer, L. Nachman, G. Stemmer, and J. Huang, "A speaker recognition approach to anomaly detection," in *5th the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 96–99.
- [27] T. Inoue, P. Vinayavekhin, S. Morikuni, S. Wang, T. H. Trong, D. Wood, M. Tatsubori, and R. Tachibana, "Detection of anomalous sounds for machine condition monitoring using classification confidence," in *5th the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2020, pp. 66–70.
- [28] Q. Zhou, "ArcFace based sound mobilenets for DCASE 2020 task 2," DCASE2020 Challenge, Tech. Rep., 2020.
- [29] K. Wilkinghoff, "Sub-cluster AdaCos: Learning representations for anomalous sound detection," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.
- [30] —, "Combining multiple distributions based on sub-cluster AdaCos for anomalous sound detection under domain shifted conditions," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2021, pp. 55–59.
- [31] S. Venkatesh, G. Wichern, A. S. Subramanian, and J. L. Roux, "Improved domain generalization via disentangled multi-task learning in unsupervised anomalous sound detection," in *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022, pp. 196–200.
- [32] T. Nishida, K. Dohi, T. Endo, M. Yamamoto, and Y. Kawaguchi, "Anomalous sound detection based on machine activity detection," in *30th European Signal Processing Conference EUSIPCO*. IEEE, 2022, pp. 269–273.
- [33] K. Wilkinghoff, "Design choices for learning embeddings from auxiliary tasks for domain generalization in anomalous sound detection," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.
- [34] D. Hendrycks, M. Mazeika, and T. G. Dietterich, "Deep anomaly detection with outlier exposure," in *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.
- [35] P. Primus, V. Hauns Schmid, P. Praher, and G. Widmer, "Anomalous sound detection as a simple binary classification problem with careful selection of proxy outlier examples," in *5th the Workshop on Detection and Classification of Acoustic Scenes and Events 2020 (DCASE)*, 2020, pp. 170–174.
- [36] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2017, pp. 6738–6746.
- [37] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 5265–5274.
- [38] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 4690–4699.
- [39] S. Grollmisch, D. Johnson, J. AbeBer, and H. Lukashevich, "IAEO3-combining OpenL3 embeddings and interpolation autoencoder for anomalous sound detection," DCASE2020 Challenge, Tech. Rep., 2020.
- [40] K. Wilkinghoff, "Using look, listen, and learn embeddings for detecting anomalous sounds in machine condition monitoring," in *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2020, pp. 215–219.
- [41] R. Müller, F. Ritz, S. Illium, and C. Linnhoff-Popien, "Acoustic anomaly detection for machine sounds based on image transfer learning," in *13th International Conference on Agents and Artificial Intelligence (ICAART)*. SCITEPRESS, 2021, pp. 49–56.
- [42] K. Wilkinghoff and F. Fritz, "On using pre-trained embeddings for detecting anomalous sounds with limited training data," in *31st European Signal Processing Conference (EUSIPCO)*. IEEE, 2023.
- [43] J. A. Lopez, G. Stemmer, P. Lopez-Meyer, P. Singh, J. A. del Hoyo Ontiveros, and H. A. Cordourier, "Ensemble of complementary anomaly detectors under domain shifted conditions," in *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2021, pp. 11–15.
- [44] I. Kuroyanagi, T. Hayashi, Y. Adachi, T. Yoshimura, K. Takeda, and T. Toda, "An ensemble approach to anomalous sound detection based on conformer-based autoencoder and binary classifier incorporated with metric learning," in *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2021, pp. 110–114.
- [45] Y. Deng, A. Jiang, Y. Duan, J. Ma, X. Chen, J. Liu, P. Fan, C. Lu, and W. Zhang, "Ensemble of multiple anomalous sound detectors," in *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022, pp. 21–25.
- [46] A. Fernandez and M. D. Plumley, "Using UMAP to inspect audio data for unsupervised anomaly detection under domain-shift conditions," in *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2021, pp. 165–169.
- [47] A. Holzinger, A. Saranti, C. Molnar, P. Biecek, and W. Samek, "Explainable AI methods - A brief overview," in *xxAI - Beyond Explainable AI - International Workshop, Held in Conjunction with ICML 2020*, ser. Lecture Notes in Computer Science, vol. 13200. Springer, 2020, pp. 13–38.
- [48] K. T. Mai, T. Davies, L. D. Griffin, and E. Benetos, "Explaining the decision of anomalous sound detectors," in *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022.
- [49] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1135–1144.

- [50] S. Mishra, B. L. Sturm, and S. Dixon, "Local interpretable model-agnostic explanations for music content analysis," in *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 537–543.
- [51] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," 2018. [Online]. Available: <https://arxiv.org/abs/1802.03426>
- [52] L. Ruff, N. Görnitz, L. Deecke, S. A. Siddiqui, R. A. Vandermeulen, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *35th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 4390–4399.
- [53] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [54] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.
- [55] P. Perera and V. M. Patel, "Learning deep features for one-class classification," *IEEE Transactions on Image Processing*, vol. 28, no. 11, pp. 5450–5463, 2019.
- [56] A. N. Gorban, I. Y. Tyukin, D. V. Prokhorov, and K. I. Sofeikov, "Approximation with random bases: Pro et contra," *Information Sciences*, vol. 364–365, pp. 129–145, 2016.
- [57] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *32nd International Conference on Machine Learning (ICML)*, vol. 37, 2015, pp. 448–456.
- [58] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *6th International Conference on Learning Representations (ICLR)*, 2018.
- [59] X. Zhang, R. Zhao, Y. Qiao, X. Wang, and H. Li, "AdaCos: Adaptively scaling cosine logits for effectively learning deep face representations," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 10 823–10 832.
- [60] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.
- [61] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 4077–4087.
- [62] J. S. Chung, J. Huh, S. Mun, M. Lee, H. Heo, S. Choe, C. Ham, S. Jung, B. Lee, and I. Han, "In defence of metric learning for speaker recognition," in *21st Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA, 2020, pp. 2977–2981.
- [63] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 63:1–63:34, 2021.
- [64] K. Wilkinghoff and A. Cornaggia-Urrigshardt, "TACos: Learning temporally structured embeddings for few-shot keyword spotting with dynamic time warping," 2023, arXiv:2305.10816.
- [65] N. Harada, D. Niizumi, D. Takeuchi, Y. Ohishi, M. Yasuda, and S. Saito, "ToyADMOS2: Another dataset of miniature-machine operating sounds for anomalous sound detection under domain shift conditions," in *6th Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. Tampere University, 2021, pp. 1–5.
- [66] J. Wang, C. Lan, C. Liu, Y. Ouyang, and T. Qin, "Generalizing to unseen domains: A survey on domain generalization," in *Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*. ijcai.org, 2021, pp. 4627–4635.
- [67] K. Dohi, K. Imoto, N. Harada, D. Niizumi, Y. Koizumi, T. Nishida, H. Purohit, R. Tanabe, T. Endo, and Y. Kawaguchi, "Description and discussion on DCASE 2023 challenge task 2: First-shot unsupervised anomalous sound detection for machine condition monitoring," in *8th Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. Tampere University, 2023, pp. 31–35.
- [68] N. Harada, D. Niizumi, D. Takeuchi, Y. Ohishi, and M. Yasuda, "First-shot anomaly detection for machine condition monitoring: A domain generalization baseline," in *31st European Signal Processing Conference (EUSIPCO)*. IEEE, 2023.
- [69] —, "ToyADMOS2+: New toyadmos data and benchmark results of the first-shot anomalous sound event detection baseline," in *8th Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. Tampere University, 2023, pp. 41–45.
- [70] D. K. McClish, "Analyzing a portion of the ROC curve," *Medical decision making*, vol. 9, no. 3, pp. 190–195, 1989.
- [71] J. Ebbers, R. Haeb-Umbach, and R. Serizel, "Threshold independent evaluation of sound event detection scores," in *International Conference on Acoustics, Speech and Signal Processing, (ICASSP)*. IEEE, 2022, pp. 1021–1025.
- [72] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.
- [73] V. Petsiuk, A. Das, and K. Saenko, "RISE: Randomized input sampling for explanation of black-box models," in *British Machine Vision Conference (BMVC)*. BMVA Press, 2018, p. 151.
- [74] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. 11, pp. 2431—2456, 2008.



Kevin Wilkinghoff received his B.Sc. degree in Mathematics at the University of Münster, Germany, and his M.Sc. degree in Computer Science at the University of Bonn, Germany, in 2014 and 2017, respectively. Since 2017 he is a research associate at Fraunhofer FKIE. Currently, he is working towards a Ph.D. degree in Computer Science at the University of Bonn. His research interests include anomaly detection, open-set classification and representation learning for machine listening applications. In 2021, he received the DCASE Best Paper Award.



Frank Kurth studied Computer Science and Mathematics at Bonn University, Germany, where he received both a masters degree in Computer Science and the degree of a doctor of natural sciences (Dr. rer. nat.) in 1997 and 1999, respectively. From 1997–2007 he was with the Multimedia Signal Processing group at Bonn University where he finished his Habilitation in Computer Science in 2004 and was subsequently appointed apl. Professor in 2013. Since 2007 he is with Fraunhofer FKIE, Germany, where he currently heads a research group focused on physical layer signal analysis in the area of communications. His research interests include the application of pattern recognition and machine learning techniques to audio, speech and communication signal processing. Dr. Kurth has received the 2000 Dissertation Award of the German Informatics Society (GI) and a 2000 Multimedia Award of the German Department of Economy and Technology. He is co-author of more than 100 publications and holds several patents. Dr. Kurth is a senior member of the IEEE.

LIST OF FIGURES

Figure 1	Histograms of anomaly scores and corresponding decision thresholds.	2
Figure 2	Illustration of different anomaly detection settings.	3
Figure 3	Illustration of an ideal embedding function for ASD.	5
Figure 4	Essential building blocks of a general ASD system based on audio embeddings	9
Figure 5	Ensuring an angular margin between classes i and j by shifting the decision boundaries.	21
Figure 6	Effects of varying the margin and the scale parameter of an angular margin loss on the relationship between angle and posterior probability.	23
Figure 7	Illustration of the three different data augmentation techniques used when applying SpecAugment.	29
Figure 8	Examples of ROC- and DET-curves.	35
Figure 9	Temporal development of different losses obtained on the DCASE2020 dataset when training by minimizing the Ada-Cos loss.	51
Figure 10	Scatter plot of normal and anomalous data belonging to two different classes.	59
Figure 11	Comparison of the AUC-ROCs and pAUCs obtained on the evaluation set with the top five highest-ranked systems submitted to the DCASE2020 Challenge task 2, the proposed approach and an ensemble.	65
Figure 12	Normalized F_1 scores obtained with different threshold estimation methods on the DCASE2020 dataset.	70
Figure 13	Optimal F_1 scores obtained on the DCASE2020 dataset when using a varying percentage of normal data samples not used for training the ASD system.	71
Figure 14	Normalized F_1 scores obtained with different threshold estimation methods on the development set of the DCASE2020 dataset when using a varying percentage of normal data samples not used for training the ASD system.	71
Figure 15	Normalized F_1 scores obtained with different threshold estimation methods on the evaluation set of the DCASE2020 dataset when using a varying percentage of normal data samples not used for training the ASD system.	72
Figure 16	Toy examples of perfectly separable anomaly score distributions, each with an AUC-ROC equal to 1.	73
Figure 17	Example of computing F_1 -EV.	73

Figure 18	Comparison of several different performance measures computed on the evaluation set of task 2 of the DCASE2023 Challenge.	77
Figure 19	Comparison of threshold-independent performance measures computed on the evaluation set of task 2 of the DCASE2023 Challenge.	77
Figure 20	Sensitivity of the bounded F_1 -EV score with respect to β_{F_1-EV}	78
Figure 21	Structure of the proposed anomalous sound detection system for domain-shifted conditions.	86
Figure 22	Log-scaled spectrograms and importance maps obtained with with RISE using and not using an auxiliary classification task.	95
Figure 23	Visualizations of the test split of the development set in the learned embedding space for different loss functions and auxiliary tasks using t-SNE.	97
Figure 24	Domain-independent performance obtained on the DCASE2023 dataset with different subspace dimensions for the AdaProj loss.	103
Figure 25	Illustration of an ASD system utilizing multiple SSL approaches.	108
Figure 26	Comparison between the performances of the presented systems and official scores of the ten top-performing systems of the DCASE2023 Challenge. For the evaluations, ensembles consisting of ten sub-systems were used.	111
Figure 27	Structure of the proposed KWS system.	123
Figure 28	Illustration of combining the embeddings belonging to different segments of a single recording.	125
Figure 29	Illustration of the TACos loss function.	126

LIST OF TABLES

Table 1	Overview of different anomaly detection settings.	2
Table 2	Advantages and disadvantages of different types of audio embeddings.	30
Table 3	Structure of the DCASE2020 ASD dataset.	47
Table 4	Modified ResNet architecture used as the baseline model.	47
Table 5	Arithmetic means of all AUC-ROCs and pAUCs obtained with different losses using different auxiliary tasks over all sections of the DCASE2020 dataset.	52
Table 6	Arithmetic means of AUC-ROCs and pAUCs for different machine types obtained with different backends.	59
Table 7	Arithmetic means of AUC-ROCs and pAUCs obtained with mixup and the sub-cluster AdaCos loss using only a single center per class.	60
Table 8	Arithmetic means of AUC-ROCs and pAUCs obtained with the sub-cluster AdaCos loss for different numbers of centers per class.	62
Table 9	Mean AUC-ROCs and pAUCs per machine type obtained with different representations.	63
Table 10	Comparison of F_1 scores obtained with different threshold estimation methods on the development set of the DCASE2020 dataset.	69
Table 11	Comparison of F_1 scores obtained with different threshold estimation methods on the evaluation set of the DCASE2020 dataset.	69
Table 12	Structure of the DCASE2022 ASD dataset.	84
Table 13	Structure of the DCASE2023 ASD dataset.	85
Table 14	Sub-network architecture for DFT feature branch.	87
Table 15	Modified ResNet architecture for STFT feature branch.	88
Table 16	Comparison between using or not using trainable cluster centers and bias terms on the DCASE2022 dataset.	89
Table 17	Comparison between different input feature representations and ways of combining them on the DCASE2022 dataset.	90
Table 18	Effect of temporal normalization in domain-shifted conditions.	91
Table 19	Comparison between different backends on the DCASE2022 dataset.	93
Table 20	Effect of the presented design choices for improving the performance in domain-shifted conditions.	94

Table 21	Mean and standard deviation of the average Euclidean distance between the t-SNE projections of each anomalous sample and the closest normal sample over five trials for different losses and using different auxiliary tasks.	98
Table 22	Harmonic means of AUC-ROCs obtained with different ways to handle the temporal dimension of pre-trained embeddings.	99
Table 23	Harmonic means of AUC-ROCs for different backends and considered embeddings.	100
Table 24	Harmonic means of AUC-ROCs for different input representations.	100
Table 25	ASD performance obtained with different loss functions on the DCASE2022 and DCASE2023 datasets.	105
Table 26	Harmonic means of AUC-ROCs and pAUCs taken over all machine IDs obtained when using different SSL approaches.	110
Table 27	Harmonic means of AUC-ROCs and pAUCs taken over all machine types obtained on the DCASE2023 dataset by modifying design choices of the proposed SSL -based system.	111
Table 28	Structure of the few-shot OSC dataset for acoustic alarm detection in domestic environments.	119
Table 29	Weighted accuracies (in percent) obtained with different systems and input representations for various openness settings and number of shots per class.	120
Table 30	Modified ResNet architecture used for extracting embeddings with temporal dimension.	124
Table 31	Event-based, micro-averaged F-score, precision and recall obtained on KWS -DailyTalk with different KWS systems.	129

LIST OF ACRONYMS

AD	anomaly detection
ASD	anomalous sound detection
AUC-ROC	area under the receiver operating characteristic curve
ASR	automatic speech recognition
CD	cosine distance
CNN	convolutional neural network
CSC	closed-set classification
CTC	connectionist temporal classification
CXE	categorical crossentropy
DBA	dynamic time warping barycenter averaging
DET	detection error tradeoff
DFT	discrete Fourier transform
DTW	dynamic time warping
EER	equal error rate
EV	expected value
FeatEx	feature exchange
FN	false negatives
FNR	false negative rate
FP	false positives
FPR	false positive rate
GAN	generative adversarial network
GDP	gamma distribution percentile
GESD	generalized extreme studentized deviate

GMM	Gaussian mixture model
HFCC	human factor cepstral coefficients
HP	histogram percentile
IQR	interquartile range
k-NN	k-nearest neighbors
KWS	keyword spotting
L3	look, listen, and learn
LDA	linear discriminant analysis
LIME	local interpretable model-agnostic explanations
LOF	local outlier factor
MADE	masked autoencoder for distribution estimation
MAD	mean absolute deviation
MFCC	Mel-frequency cepstral coefficient
MSE	mean squared error
MST	multi-stage thresholding
OCSVM	one-class support vector machine
OE	outlier exposure
OSC	open-set classification
PANN	pre-trained audio neural network
pAUC	partial area under the receiver operating characteristic curve
PCA	principal component analysis
PCC	Pearson correlation coefficient
PLDA	probabilistic linear discriminant analysis
PSD	polyphonic sound detection
ReLU	rectified linear unit
RISE	randomized input sampling for explanation

ROC	receiver operating characteristic
SD	standard deviation
SED	sound event detection
SMOTE	synthetic minority over-sampling technique
SSL	self-supervised learning
StatEx	statistics exchange
STFT	short-time Fourier transform
SVDD	support vector data description
TACos	temporal AdaCos
TMN	temporal mean normalization
TN	true negatives
TNR	true negative rate
TP	true positives
TPR	true positive rate
t-SNE	t-distributed stochastic neighbor embedding
UMAP	uniform manifold approximation and projection
VAD	voice activity detection
VAE	variational autoencoder
xAI	explainable artificial intelligence

LIST OF SYMBOLS

$\mathbb{1}_I$	characteristic function for the interval I 125
\cdot	linear operator 17
α	angle between two embeddings on the hypersphere 20
$\hat{\alpha}_{\text{med}}^{(t)}$	median of all mixed-up angles and all class centers at training step t 54
$\alpha_{\text{med}}^{(t)}$	median of all angles belonging to the training samples and their corresponding class centers at training step t 23
β_{acc}	accuracy metric weight 36
$\beta_{F_1\text{-EV}}$	hyperparameter for $F_1\text{-EV}$ 75
λ	mixing coefficient of mixup 27
Λ	space of all categorical class label functions 20
ϕ	neural network for obtaining embeddings 16
Φ	space of permissible neural network architectures for obtaining embeddings 8
Φ_{simple}	space of permissible neural network architectures only consisting of fully-connected or convolutional layers 16
φ	autoencoder 14
φ_d	decoder 14
φ_e	encoder 14
σ_l	activation function for layer l 17
θ	decision threshold 1
A_{pred}	predicted anomalies 33
$B_{\text{avg}}^{(t)}$	sample-wise average over all summed logits of the AdaCos loss belonging to the non-corresponding classes at training step t 23
Beta	beta distribution 27
$\hat{B}_{\text{avg}}^{(t)}$	sample-wise average over all summed logits of the sub-cluster AdaCos loss at training step t 54
b_l	bias term of layer l 17

c	center of a hypersphere, class center, prototype 16, 20, 117
C_j	all hypersphere centers belonging to class j 53
class	class index function 8
$C_{\text{pred}}^{(i)}$	all data samples classified as class i 36
d	metric 117
D	embedding dimension 5
d_{proj}	metric measuring distance between embedding and its projection onto a linear span 102
E_i	subset of the embedding space containing only normal samples 5
emb	ideal embedding function 5
F	frequency dimension 106
Grubbs	Grubbs statistic 41
H_l	hidden representation space for layer l 16
$h_l(x)$	hidden representation at layer l for input sample x 17
I_{active}	interval of active positions 125
Im	image of a function 17
i_{pos}	index of an encoded relative position 125
i_{seg}	index of a segment 125
K	number of shots for few-shot learning 117
\mathcal{L}_{ada}	AdaCos loss 24
\mathcal{L}_{ang}	angular margin loss, ArcFace 20
$\mathcal{L}_{\text{comp}}$	compactness loss 16
\mathcal{L}_{kw}	keyword classification loss 127
\mathcal{L}_{pos}	position classification loss 127
$\mathcal{L}_{\text{proj}}$	AdaProj loss 102
$\mathcal{L}_{\text{prot}}^{(t)}$	prototypical loss at training step t 117
\mathcal{L}_{rec}	reconstruction loss 14
$\mathcal{L}_{\text{sc-ada}}$	sub-cluster AdaCos loss 54
\mathcal{L}_{tac}	TACos loss 127
lab	categorical class label function 8
lab _{pos}	categorical encoding of the relative position 125
$L(\phi)$	number of layers for neural network ϕ 16
L_{seg}	segment length of input waveforms in seconds 123
m	angular margin 20

mean	arithmetic mean of a dataset 40
median	median of a dataset 41
mix_{lab}	label mixing function of mixup 28
mix_x	sample mixing function of mixup 28
N_{batch}	batch size 23
N_{centers}	number of centers for a single class 53
N_{classes}	number of classes 8
$N_{\text{classes}}^{\text{test}}$	number of test classes 116
$N_{\text{classes}}^{\text{train}}$	number of classes used for training 116
N_{kw}	number of keyword classes 126
N_{pos}	maximum number of segments belonging to any training sample 125
N_{seg}	number of segments belonging to a training sample 125
openness	openness of an open-set task 116
\mathcal{P}	power set 8
$\text{pred}_{\text{class}}$	classifier function 36
$\mathbf{P}_{\mathcal{S}^{\mathcal{D}-1}}$	projection onto \mathcal{D} -sphere 49
$\mathbf{P}_{\text{span}(\mathbf{C})}$	projection onto the linear span of \mathbf{C} 102
$\mathcal{Q}^{(t)}$	query set at training step t 117
\mathbb{R}_+	positive real numbers including zero 14
s	scale parameter 20
$\mathcal{S}^{(t)}$	support set at training step t 117
score	anomaly score function 1
$\text{score}_{\text{emb}}$	anomaly score function in the embedding space 26
$\mathcal{S}^{\mathcal{D}-1}$	\mathcal{D} -sphere, unit sphere 49
$\hat{s}^{(t)}$	dynamically adaptive scale parameter of the sub-cluster AdaCos loss at training step t 53
$\text{shift}_{\text{domain}}$	domain shift 81
sim	cosine similarity 20
sim_{mar}	cosine similarity with angular margin 20
$\text{sim}_{\text{max}}^{(t)}$	stability term of the sub-cluster AdaCos loss at training step t 54
softmax	softmax function 20
sort	sort function sorting real values from low to high 35

$\text{span}(\mathbf{C})$	linear span of elements in \mathbf{C} 102
std	standard deviation of a dataset 40
$\tilde{\sigma}$	dynamically adaptive scale parameter 24
$\tilde{\sigma}^{(t)}$	dynamically adaptive scale parameter of the Ada-Cos loss at training step t 23
t	training step 23
T	time dimension 106
\mathcal{U}	normal distribution 28
\mathcal{W}	parameter space of a neural network 8
w_0	parameter setting with all weights equal to zero 17
$w(\mathbf{l})$	weight parameters of layer \mathbf{l} (excluding bias term) 17
x	a data sample 1
\mathcal{X}	data space containing raw waveforms or pre-processed feature representations 1, 13
$\mathcal{X}_{\text{anomalous}}$	anomalous data samples 1
$\mathcal{X}_{\text{normal}}$	normal data samples 1
$\mathcal{X}_{\text{normal}}^{(i)}$	normal data samples belonging to class i 36
$\mathcal{X}_{\text{source}}$	source domain 81
$\mathcal{X}_{\text{target}}$	target domain 81
$\mathcal{X}_{\text{train}}$	training samples 2
\mathcal{Y}	finite subset of the data space 14
\mathcal{Y}_j	subset of the data space containing only samples of class j 55
$\mathcal{Y}^{(t)}$	batch at training step t 23

BIBLIOGRAPHY

- [1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. “Tensorflow: A system for large-scale machine learning.” In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 2016, pp. 265–283.
- [2] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. “YouTube-8M: A Large-Scale Video Classification Benchmark.” In: *CoRR* abs/1609.08675 (2016).
- [3] Charu Aggarwal. *Outlier Analysis*. 2nd. Springer, 2017.
- [4] Samet Akcay, Amir Atapour Abarghouei, and Toby P. Breckon. “GANomaly: Semi-supervised Anomaly Detection via Adversarial Training.” In: *14th Asian Conference on Computer Vision (ACCV)*. Vol. 11363. Lecture Notes in Computer Science. Springer, 2018, pp. 622–637.
- [5] Guillaume Alain and Yoshua Bengio. “What regularized auto-encoders learn from the data-generating distribution.” In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 3563–3593.
- [6] Relja Arandjelovic and Andrew Zisserman. “Look, Listen and Learn.” In: *International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2017, pp. 609–617.
- [7] Relja Arandjelovic and Andrew Zisserman. “Objects that Sound.” In: *15th European Conference on Computer Vision (ECCV)*. Vol. 11205. Lecture Notes in Computer Science. Springer, 2018, pp. 451–466.
- [8] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations.” In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2020.
- [9] Jisheng Bai, Jianfeng Chen, Mou Wang, Muhammad Saad Ayub, and Qingli Yan. “SSDPT: Self-supervised dual-path transformer for anomalous sound detection.” In: *Digit. Signal Process.* 135 (2023), p. 103939.
- [10] Jisheng Bai, Yafei Jia, and Siwei Huang. *JLESS Submission to DCASE2022 Task2: Batch Mixing Strategy Based Method With Anomaly Detector for Anomalous Sound Detection*. Tech. rep. DCASE Challenge, 2022.

- [11] Jisheng Bai, Mou Wang, and Jianfeng Chen. “Dual-Path Transformer For Machine Condition Monitoring.” In: *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE, 2021, pp. 1144–1148.
- [12] Adrien Bardes, Jean Ponce, and Yann LeCun. “VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning.” In: *10th International Conference on Learning Representations ICLR*. Open-Review.net, 2022.
- [13] Yacine Bel-Hadj and Wout Weijtjens. “Population-Based SHM Under Environmental Variability Using a Classifier for Unsupervised Damage Detection.” In: *14th International Workshop on Structural Health Monitoring*. 2023, pp. 1479–1488.
- [14] Yacine Bel-Hadj, Wout Weijtjens, and Francisco de Nolasco Santos. “Anomaly detection and representation learning in an instrumented railway bridge.” In: *30th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*. 2022.
- [15] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. “Analysis of Representations for Domain Adaptation.” In: *Advances in Neural Information Processing Systems 19: Twentieth Annual Conference on Neural Information Processing Systems (NIPS)*. MIT Press, 2006, pp. 137–144.
- [16] Çağdas Bilen, Giacomo Ferroni, Francesco Tuveri, Juan Azcarreta, and Sacha Krstulovic. “A Framework for the Robust Evaluation of Sound Event Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 61–65.
- [17] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. “Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models.” In: *IEEE Trans. Pattern Anal. Mach. Intell.* 44.11 (2022), pp. 7327–7347.
- [18] Holger Severin Bovbjerg and Zheng-Hua Tan. “Improving Label-Deficient Keyword Spotting Through Self-Supervised Pretraining.” In: *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2023.
- [19] Andrew P. Bradley. “The use of the area under the ROC curve in the evaluation of machine learning algorithms.” In: *Pattern recognition* 30.7 (1997), pp. 1145–1159.
- [20] Guido Buzzi-Ferraris and Flavio Manenti. “Outlier detection in large data sets.” In: *Computers & chemical engineering* 35.2 (2011), pp. 388–390.

- [21] Xinyu Cai, Heinrich Dinkel, Zhiyong Yan, Yongqing Wang, Junbo Zhang, Zhiyong Wu, and Yujun Wang. “A Contrastive Semi-Supervised Learning Framework For Anomaly Sound Detection.” In: *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. 2021, pp. 31–34.
- [22] Estefanía Cano, Johannes Nowak, and Sascha Grollmisch. “Exploring sound source separation for acoustic condition monitoring in industrial scenarios.” In: *25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 2264–2268.
- [23] Tymoteusz Cejrowski and Julian Szymanski. “Buzz-based honeybee colony fingerprint.” In: *Comput. Electron. Agric.* 191 (2021), p. 106489.
- [24] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. “SMOTE: Synthetic Minority Over-sampling Technique.” In: *J. Artif. Intell. Res.* 16 (2002), pp. 321–357.
- [25] Han Chen, Yan Song, Li-Rong Dai, Ian McLoughlin, and Lin Liu. “Self-Supervised Representation Learning for Unsupervised Anomalous Sound Detection Under Domain Shift.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 471–475.
- [26] Han Chen, Yan Song, Zhu Zhuo, Yu Zhou, Yu-Hong Li, Hui Xue, and Ian McLoughlin. “An Effective Anomalous Sound Detection Method Based on Representation Learning with Simulated Anomalies.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.
- [27] Rewon Child. “Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images.” In: *9th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021.
- [28] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. “On the properties of neural machine translation: Encoder–decoder approaches.” In: *8th Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST)*. Association for Computational Linguistics (ACL). 2014, pp. 103–111.
- [29] Kateryna Chumachenko, Alexandros Iosifidis, and Moncef Gabbouj. “Robust Fast Subclass Discriminant Analysis.” In: *28th European Signal Processing Conference (EUSIPCO)*. IEEE. 2020, pp. 1397–1401.
- [30] Joon Son Chung, Jaesung Huh, Seongkyu Mun, Minjae Lee, Hee-Soo Heo, Soyeon Choe, Chiheon Ham, Sunghwan Jung, Bong-Jin Lee, and Icksang Han. “In Defence of Metric Learning for Speaker Recognition.” In: *21st Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA, 2020, pp. 2977–2981.

- [31] James Clark, Zhen Liu, and Nathalie Japkowicz. “Adaptive Threshold for Outlier Detection on Data Streams.” In: *5th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2018, pp. 41–49.
- [32] Aurora Cramer, Ho-Hsiang Wu, Justin Salamon, and Juan Pablo Bello. “Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3852–3856.
- [33] Pawel Daniluk, Marcin Gozdziowski, Slawomir Kapka, and Michal Kosmider. *Ensemble of Auto-Encoder Based Systems for Anomaly Detection*. Tech. rep. DCASE2020 Challenge, 2020.
- [34] Ingrid Daubechies. *Ten Lectures on Wavelets*. SIAM, 1992.
- [35] Jesse Davis and Mark Goadrich. “The relationship between Precision-Recall and ROC curves.” In: *Twenty-Third International Conference on Machine Learning (ICML)*. Vol. 148. ACM International Conference Proceeding Series. ACM, 2006, pp. 233–240.
- [36] Steven Davis and Paul Mermelstein. “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences.” In: *IEEE Trans. Acoust. Speech Signal Process* 28.4 (1980), pp. 357–366.
- [37] Jiankang Deng, Jia Guo, Tongliang Liu, Mingming Gong, and Stefanos Zafeiriou. “Sub-center ArcFace: Boosting face recognition by large-scale noisy web faces.” In: *European Conference on Computer Vision (ECCV)*. Springer. 2020, pp. 741–757.
- [38] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. “ArcFace: Additive Angular Margin Loss for Deep Face Recognition.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 4690–4699.
- [39] Yufeng Deng, Anbai Jiang, Yuchen Duan, Jitao Ma, Xuchu Chen, Jia Liu, Pingyi Fan, Cheng Lu, and Wei-Qiang Zhang. “Ensemble of Multiple Anomalous Sound Detectors.” In: *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022.
- [40] Theekshana Dissanayake, Tharindu Fernando, Simon Denman, Sridha Sridharan, Houman Ghaemmaghani, and Clinton Fookes. “A Robust Interpretable Deep Learning Classifier for Heart Anomaly Detection Without Segmentation.” In: *IEEE J. Biomed. Health Informatics* 25.6 (2021), pp. 2162–2171.
- [41] Duygu Dogan, Huang Xie, Toni Heittola, and Tuomas Virtanen. “Zero-Shot Audio Classification using Image Embeddings.” In: *30th European Signal Processing Conference (EUSIPCO)*. IEEE, 2022, pp. 1–5.

- [42] Kota Dohi, Takashi Endo, and Yohei Kawaguchi. “Disentangling physical parameters for anomalous sound detection under domain shifts.” In: *30th European Signal Processing Conference (EUSIPCO)*. IEEE, 2022, pp. 279–283.
- [43] Kota Dohi, Takashi Endo, Harsh Purohit, Ryo Tanabe, and Yohei Kawaguchi. “Flow-Based Self-Supervised Density Estimation for Anomalous Sound Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 336–340.
- [44] Kota Dohi, Keisuke Imoto, Noboru Harada, Daisuke Niizumi, Yuma Koizumi, Tomoya Nishida, Harsh Purohit, Ryo Tanabe, Takashi Endo, and Yohei Kawaguchi. “Description and Discussion on DCASE 2023 Challenge Task 2: First-Shot Unsupervised Anomalous Sound Detection for Machine Condition Monitoring.” In: *8th Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. Tampere, Finland, 2023, pp. 31–35.
- [45] Kota Dohi, Tomoya Nishida, Harsh Purohit, Ryo Tanabe, Takashi Endo, Masaaki Yamamoto, Yuki Nikaido, and Yohei Kawaguchi. “MIMII DG: Sound Dataset for Malfunctioning Industrial Machine Investigation and Inspection for Domain Generalization Task.” In: *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022, pp. 1–5.
- [46] Kota Dohi et al. “Description and Discussion on DCASE 2022 Challenge Task 2: Unsupervised Anomalous Sound Detection for Machine Condition Monitoring Applying Domain Generalization Techniques.” In: *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022, pp. 26–30.
- [47] Jiaxin Du, Guangjie Han, Chuan Lin, and Miguel Martínez-García. “ITrust: An Anomaly-Resilient Trust Model Based on Isolation Forest for Underwater Acoustic Sensor Networks.” In: *IEEE Trans. Mob. Comput.* 21.5 (2022), pp. 1684–1696.
- [48] Janek Ebberts, Reinhold Haeb-Umbach, and Romain Serizel. “Threshold Independent Evaluation of Sound Event Detection Scores.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 1021–1025.
- [49] Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. “CLAP Learning Audio Concepts from Natural Language Supervision.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.

- [50] Andres Fernandez and Mark D. Plumbley. “Using UMAP to Inspect Audio Data for Unsupervised Anomaly Detection Under Domain-Shift Conditions.” In: *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. 2021, pp. 165–169.
- [51] Giacomo Ferroni, Nicolas Turpault, Juan Azcarreta, Francesco Tuveri, Romain Serizel, Çağdas Bilen, and Sacha Krstulovic. “Improving Sound Event Detection Metrics: Insights from DCASE 2020.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 631–635.
- [52] Peter Filzmoser. “A multivariate outlier detection method.” In: *7th International Conference on Computer Data Analysis and Modeling*. 2004, pp. 18–22.
- [53] Pasquale Foggia, Nicolai Petkov, Alessia Saggese, Nicola Strisciuglio, and Mario Vento. “Audio Surveillance of Roads: A System for Detecting Anomalous Sounds.” In: *IEEE Trans. Intell. Transp. Syst.* 17.1 (2016), pp. 279–288.
- [54] M. A. Ganaie, Minghui Hu, Ashwani Kumar Malik, Muhammad Tanveer, and Ponnuthurai N. Suganthan. “Ensemble deep learning: A review.” In: *Eng. Appl. Artif. Intell.* 115 (2022), p. 105151.
- [55] Jing Gao and Pang-Ning Tan. “Converting Output Scores from Outlier Detection Algorithms into Probability Estimates.” In: *6th International Conference on Data Mining (ICDM)*. IEEE Computer Society, 2006, pp. 212–221.
- [56] Martin Gebel. “Multivariate calibration of classifier scores into the probability space.” Ph.D. thesis. University of Dortmund, 2009.
- [57] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. “Audio Set: An ontology and human-labeled dataset for audio events.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.
- [58] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. “MADE: Masked Autoencoder for Distribution Estimation.” In: *32nd International Conference on Machine Learning (ICML)*. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, 2015, pp. 881–889.
- [59] Ritwik Giri, Fangzhou Cheng, Karim Helwani, Srikanth V. Tenneti, Umut Isik, and Arvinhd Krishnaswamy. “Group Masked Autoencoder Based Density Estimator for Audio Anomaly Detection.” In: *5th the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. 2020, pp. 51–55.

- [60] Ritwik Giri, Srikanth V. Tenneti, Fangzhou Cheng, Karim Helwani, Umut Isik, and Arvindh Krishnaswamy. “Self-Supervised Classification for Detecting Anomalous Sounds.” In: *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. 2020, pp. 46–50.
- [61] Ritwik Giri, Srikanth V. Tenneti, Karim Helwani, Fangzhou Cheng, Umut Isik, and Arvindh Krishnaswamy. *Unsupervised Anomalous Sound Detection Using Self-Supervised Classification and Group Masked Autoencoder for Density Estimation*. Tech. rep. DCASE2020 Challenge, 2020.
- [62] Tobias Glasmachers. “Limits of End-to-End Learning.” In: *9th Asian Conference on Machine Learning, ACML*. Vol. 77. Proceedings of Machine Learning Research. PMLR, 2017, pp. 17–32.
- [63] Kaan Gökcesu, Mohammadreza Mohaghegh Neyshabouri, Hakan Gökcesu, and Suleyman Serdar Kozat. “Sequential Outlier Detection Based on Incremental Decision Trees.” In: *IEEE Trans. Signal Process.* 67.4 (2019), pp. 993–1005.
- [64] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [65] Alexander N. Gorban, Ivan Yu. Tyukin, Danil V. Prokhorov, and Konstantin I. Sofeikov. “Approximation with random bases: Pro et Contra.” In: *Information Sciences* 364-365 (2016), pp. 129–145.
- [66] Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks.” In: *23rd International Conference on Machine Learning (ICML)*. ACM, 2006, pp. 369–376.
- [67] Sascha Grollmisch, Estefanía Cano, Christian Kehling, and Michael Taenzer. “Analyzing the Potential of Pre-Trained Embeddings for Audio Classification Tasks.” In: *28th European Signal Processing Conference (EUSIPCO)*. IEEE, 2020, pp. 790–794.
- [68] Sascha Grollmisch, David Johnson, Jakob Abeßer, and Hanna Lukashevich. “IAEO3-combining OpenL3 embeddings and interpolation autoencoder for anomalous sound detection.” In: *Tech. Rep., DCASE2020 Challenge* (2020).
- [69] Frank E. Grubbs. “Sample Criteria for Testing Outlying Observations.” In: *The Annals of Mathematical Statistics* 21.1 (1950), pp. 27–58.
- [70] Jian Guan, Youde Liu, Qiaoxi Zhu, Tieran Zheng, Jiqing Han, and Wenwu Wang. “Time-Weighted Frequency Domain Audio Representation with GMM Estimator for Anomalous Sound Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.
- [71] James A. Hanley and Barbara J. McNeil. “The meaning and use of the area under a receiver operating characteristic (ROC) curve.” In: *Radiology* 143.1 (1982), pp. 29–36.

- [72] Noboru Harada, Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, and Masahiro Yasuda. “First-Shot Anomaly Detection for Machine Condition Monitoring: A Domain Generalization Baseline.” In: *31st European Signal Processing Conference (EUSIPCO)*. IEEE, 2023, pp. 191–195.
- [73] Noboru Harada, Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, and Masahiro Yasuda. “ToyADMOS2+: New Toyadmos Data and Benchmark Results of the First-Shot Anomalous Sound Event Detection Baseline.” In: *8th Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. Tampere University, 2023, pp. 41–45.
- [74] Noboru Harada, Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Masahiro Yasuda, and Shoichiro Saito. “ToyADMOS2: Another Dataset of Miniature-Machine Operating Sounds for Anomalous Sound Detection under Domain Shift Conditions.” In: *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. 2021, pp. 1–5.
- [75] Tomoki Hayashi, Tatsuya Komatsu, Reishi Kondo, Tomoki Toda, and Kazuya Takeda. “Anomalous Sound Event Detection Based on WaveNet.” In: *26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 2494–2498.
- [76] Tomoki Hayashi, Takenori Yoshimura, and Yusuke Adachi. *Conformer-Based ID-Aware Autoencoder for Unsupervised Anomalous Sound Detection*. Tech. rep. DCASE2020 Challenge, 2020.
- [77] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778.
- [78] Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. “Deep Anomaly Detection with Outlier Exposure.” In: *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.
- [79] Hynek Hermansky. “Perceptual linear predictive (PLP) analysis of speech.” In: *The Journal of the Acoustical Society of America* 87.4 (1990), pp. 1738–1752.
- [80] Shawn Hershey et al. “CNN architectures for large-scale audio classification.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 131–135.
- [81] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. “Improving neural networks by preventing co-adaptation of feature detectors.” In: *CoRR* abs/1207.0580 (2012).
- [82] Geoffrey Everest Hinton and Ruslan Salakhutdinov. “Reducing the Dimensionality of Data with Neural Networks.” In: *Science* 313.5786 (2006), pp. 504–507.

- [83] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory.” In: *Neural computation* 9.8 (1997), pp. 1735–1780.
- [84] Andreas Holzinger, Anna Saranti, Christoph Molnar, Przemyslaw Biecek, and Wojciech Samek. “Explainable AI Methods - A Brief Overview.” In: *xxAI - Beyond Explainable AI - International Workshop, Held in Conjunction with ICML 2020*. Vol. 13200. Lecture Notes in Computer Science. Springer, 2020, pp. 13–38.
- [85] Tadanobu Inoue, Phongtharin Vinayavekhin, Shu Morikuni, Shiqiang Wang, Tuan Hoang Trong, David Wood, Michiaki Tatsubori, and Ryuki Tachibana. “Detection of Anomalous Sounds for Machine Condition Monitoring using Classification Confidence.” In: *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. 2020, pp. 66–70.
- [86] Sergey Ioffe and Christian Szegedy. “Batch normalization: accelerating deep network training by reducing internal covariate shift.” In: *32nd International Conference on Machine Learning (ICML)*. Vol. 37. 2015, pp. 448–456.
- [87] Alan Jeffares, Qinghai Guo, Pontus Stenetorp, and Timoleon Moraitis. “Spike-inspired rank coding for fast and accurate recurrent neural networks.” In: *10th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2022.
- [88] Wang JiaJun. *Self-Supervised Representation Learning for First-Shot Unsupervised Anomalous Sound Detection*. Tech. rep. DCASE2023 Challenge, June 2023.
- [89] Anbai Jiang, Qijun Hou, Jia Liu, Pingyi Fan, Jitao Ma, Cheng Lu, Yuanzhi Zhai, Yufeng Deng, and Wei-Qiang Zhang. *THUEE System for First-Shot Unsupervised Anomalous Sound Detection for Machine Condition Monitoring*. Tech. rep. DCASE2023 Challenge, 2023.
- [90] Anbai Jiang, Wei-Qiang Zhang, Yufeng Deng, Pingyi Fan, and Jia Liu. “Unsupervised Anomaly Detection and Localization of Machine Audio: A GAN-Based Approach.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.
- [91] Yulei Jiang, Charles E. Metz, and Robert M. Nishikawa. “A receiver operating characteristic partial area index for highly sensitive diagnostic tests.” In: *Radiology* 201.3 (1996), pp. 745–750.
- [92] Justin M. Johnson and Taghi M. Khoshgoftaar. “Survey on deep learning with class imbalance.” In: *J. Big Data* 6 (2019), p. 27.
- [93] Wang Junjie, Wang Jiajun, Chen Shengbing, Sun Yong, and Liu Mengyuan. *Anomalous Sound Detection Based on Self-Supervised Learning*. Tech. rep. DCASE2023 Challenge, 2023.

- [94] Herman Kamper, Weiran Wang, and Karen Livescu. “Deep convolutional acoustic word embeddings using word-pair side information.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4950–4954.
- [95] Slawomir Kapka. “ID-Conditioned Auto-Encoder for Unsupervised Anomaly Detection.” In: *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. 2020, pp. 71–75.
- [96] Yohei Kawaguchi, Keisuke Imoto, Yuma Koizumi, Noboru Harada, Daisuke Niizumi, Kota Dohi, Ryo Tanabe, Harsh Purohit, and Takashi Endo. “Description and Discussion on DCASE 2021 Challenge Task 2: Unsupervised Anomalous Detection for Machine Condition Monitoring Under Domain Shifted Conditions.” In: *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. 2021, pp. 186–190.
- [97] Byeonggeun Kim, Mingu Lee, Jinkyu Lee, Yeonseok Kim, and Kyuwoong Hwang. “Query-by-Example On-Device Keyword Spotting.” In: *Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 532–538.
- [98] Byeonggeun Kim, Seunghan Yang, Inseop Chung, and Simyung Chang. “Dummy Prototypical Networks for Few-Shot Open-Set Keyword Spotting.” In: *23rd Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA, 2022, pp. 4621–4625.
- [99] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, S. M. Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. “Attentive Neural Processes.” In: *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.
- [100] Jaechang Kim, Yunjoo Lee, Hyun Mi Cho, Dong Woo Kim, Chi Hoon Song, and Jungseul Ok. “Activity-Informed Industrial Audio Anomaly Detection Via Source Separation.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.
- [101] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” In: *3rd International Conference on Learning Representations (ICLR)*. 2015.
- [102] Diederik P. Kingma and Max Welling. “An Introduction to Variational Autoencoders.” In: *Found. Trends Mach. Learn.* 12.4 (2019), pp. 307–392.
- [103] R. Kirandevraj, Vinod Kumar Kurmi, Vinay P. Namboodiri, and C. V. Jawahar. “Generalized Keyword Spotting using ASR embeddings.” In: *23rd Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA, 2022, pp. 126–130.

- [104] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. “Why Normalizing Flows Fail to Detect Out-of-Distribution Data.” In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2020.
- [105] Ivan Kobyzev, Simon J. D. Prince, and Marcus A. Brubaker. “Normalizing Flows: An Introduction and Review of Current Methods.” In: *IEEE Trans. Pattern Anal. Mach. Intell.* 43.11 (2021), pp. 3964–3979.
- [106] Yuma Koizumi, Shoichiro Saito, Hisashi Uematsu, Noboru Harada, and Keisuke Imoto. “ToyADMOS: A dataset of miniature-machine operating sounds for anomalous sound detection.” In: *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 313–317.
- [107] Yuma Koizumi, Shoichiro Saito, Hisashi Uematsu, Yuta Kawachi, and Noboru Harada. “Unsupervised Detection of Anomalous Sound Based on Deep Learning and the Neyman-Pearson Lemma.” In: *IEEE ACM Trans. Audio Speech Lang. Process.* 27.1 (2019), pp. 212–224.
- [108] Yuma Koizumi et al. “Description and Discussion on DCASE2020 Challenge Task2: Unsupervised Anomalous Sound Detection for Machine Condition Monitoring.” In: *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. 2020, pp. 81–85.
- [109] Alexander Kolesnikov and Christoph H. Lampert. “Seed, Expand and Constrain: Three Principles for Weakly-Supervised Image Segmentation.” In: *14th European Conference on Computer Vision (ECCV)*. Vol. 9908. Lecture Notes in Computer Science. Springer, 2016, pp. 695–711.
- [110] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D. Plumbley. “PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition.” In: *IEEE ACM Trans. Audio Speech Lang. Process.* 28 (2020), pp. 2880–2894.
- [111] Anurag Kumar, Maksim Khadkevich, and Christian Fügen. “Knowledge Transfer from Weakly Labeled Audio Using Convolutional Neural Network for Sound Events and Scenes.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 326–330.
- [112] Ibuki Kuroyanagi, Tomoki Hayashi, Yusuke Adachi, Takenori Yoshimura, Kazuya Takeda, and Tomoki Toda. “An Ensemble Approach to Anomalous Sound Detection Based on Conformer-Based Autoencoder and Binary Classifier Incorporated with Metric Learning.” In: *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. 2021, pp. 110–114.

- [113] Ibuki Kuroyanagi, Tomoki Hayashi, Kazuya Takeda, and Tomoki Toda. “Anomalous Sound Detection Using a Binary Classification Model and Class Centroids.” In: *29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 1995–1999.
- [114] Ibuki Kuroyanagi, Tomoki Hayashi, Kazuya Takeda, and Tomoki Toda. “Improvement of Serial Approach to Anomalous Sound Detection by Incorporating Two Binary Cross-Entropies for Outlier Exposure.” In: *30th European Signal Processing Conference (EUSIPCO)*. IEEE, 2022, pp. 294–298.
- [115] Ibuki Kuroyanagi, Tomoki Hayashi, Kazuya Takeda, and Tomoki Toda. *Two-stage anomalous sound detection systems using domain generalization and specialization techniques*. Tech. rep. DCASE Challenge, 2022.
- [116] Frank Kurth and Dirk von Zeddelmann. “An Analysis of MFCC-like Parametric Audio Features for Keyphrase Spotting Applications.” In: *ITG Symposium on Speech Communication (ITG Speech)*. VDE, 2010.
- [117] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. “Zero-data Learning of New Tasks.” In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2008, pp. 646–651.
- [118] Keon Lee, Kyumin Park, and Daeyoung Kim. “DailyTalk: Spoken Dialogue Dataset for Conversational Text-to-Speech.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.
- [119] Guillaume Lemaitre, Fernando Nogueira, and Christos K. Aridas. “Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning.” In: *J. Mach. Learn. Res.* 18 (2017), 17:1–17:5.
- [120] Bin Li and Emmanuel Müller. “Contrastive Time Series Anomaly Detection by Temporal Transformations.” In: *International Joint Conference on Neural Networks, (IJCNN)*. IEEE, 2023.
- [121] Haoyuan Li and Yifan Li. “Anomaly detection methods based on GAN: a survey.” In: *Appl. Intell.* 53.7 (2023), pp. 8209–8231.
- [122] Jinyu Li et al. “Recent advances in end-to-end automatic speech recognition.” In: *APSIPA Transactions on Signal and Information Processing* 11.1 (2022).
- [123] Kai Li, Quoc-Huy Nguyen, Yasuji Ota, and Masashi Unoki. “Unsupervised Anomalous Sound Detection for Machine Condition Monitoring Using Temporal Modulation Features on Gammatone Auditory Filterbank.” In: *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022.

- [124] Kai Li, Dung Kim Tran, Xugang Lu, Masato Akagi, and Masashi Unoki. “Data-driven Non-uniform Filterbanks Based on F-ratio for Machine Anomalous Sound Detection.” In: *31st European Signal Processing Conference (EUSIPCO)*. IEEE, 2023, pp. 201–205.
- [125] Lijian Li, Yunhe Zhang, and Aiping Huang. “Learnable Subspace Orthogonal Projection for Semi-supervised Image Classification.” In: *16th Asian Conference on Computer Vision (ACCV)*. Vol. 13843. Lecture Notes in Computer Science. Springer, 2022, pp. 477–490.
- [126] Renjie Li, Xiaohua Gu, Fei Lu, Hongfei Song, and Jutao Pan. *Unsupervised Adversarial domain adaptive abnormal sound detection for machine condition monitoring under Domain Shift Conditions*. Tech. rep. DCASE2021 Challenge, 2021.
- [127] Yanxiong Li, Xianku Li, Yuhan Zhang, Mingle Liu, and Wucheng Wang. “Anomalous Sound Detection Using Deep Audio Representation and a BLSTM Network for Audio Surveillance of Roads.” In: *IEEE Access* 6 (2018), pp. 58043–58055.
- [128] Min Lin, Qiang Chen, and Shuicheng Yan. “Network In Network.” In: *2nd International Conference on Learning Representations, (ICLR)*. Ed. by Yoshua Bengio and Yann LeCun. 2014.
- [129] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. “Focal Loss for Dense Object Detection.” In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2017, pp. 2999–3007.
- [130] Yihan Lin, Xunquan Chen, Ryoichi Takashima, and Tetsuya Takiguchi. “Zero-Shot Sound Event Classification Using a Sound Attribute Vector with Global and Local Feature Learning.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.
- [131] Haohe Liu, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Wenwu Wang, and Mark D. Plumbley. “Segment-Level Metric Learning for Few-Shot Bioacoustic Event Detection.” In: *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022.
- [132] Shuo Liu, Adria Mallol-Ragolta, Emilia Parada-Cabaleiro, Kun Qian, Xin Jing, Alexander Kathan, Bin Hu, and Björn W. Schuller. “Audio self-supervised learning: A survey.” In: *Patterns* 3.12 (2022), p. 100616.
- [133] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. “SphereFace: Deep Hypersphere Embedding for Face Recognition.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2017, pp. 6738–6746.

- [134] Jose A. Lopez, Hong Lu, Paulo Lopez-Meyer, Lama Nachman, Georg Stemmer, and Jonathan Huang. “A Speaker Recognition Approach to Anomaly Detection.” In: *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. 2020, pp. 96–99.
- [135] Jose A. Lopez, Georg Stemmer, Paulo Lopez-Meyer, Pradyumna Singh, Juan A. del Hoyo Ontiveros, and Héctor A. Cordourier. “Ensemble Of Complementary Anomaly Detectors Under Domain Shifted Conditions.” In: *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. 2021, pp. 11–15.
- [136] Iván López-Espejo, Zheng-Hua Tan, John H. L. Hansen, and Jesper Jensen. “Deep Spoken Keyword Spotting: An Overview.” In: *IEEE Access* 10 (2022), pp. 4169–4199.
- [137] Zhiqiang Lv, Bing Han, Zhengyang Chen, Yanmin Qian, Jiawei Ding, and Jia Liu. *Unsupervised Anomalous Detection Based on Unsupervised Pre-trained Models*. Tech. rep. DCASE2023 Challenge, 2023.
- [138] Haoxin Ma, Ye Bai, Jiangyan Yi, and Jianhua Tao. “Hypersphere Embedding and Additive Margin for Query-by-example Keyword Spotting.” In: *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2019, pp. 868–872.
- [139] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. “Rectifier nonlinearities improve neural network acoustic models.” In: *30th International Conference on Machine Learning (ICML)*. 2013.
- [140] Kimberly T. Mai, Toby Davies, Lewis D. Griffin, and Emmanouil Benetos. “Explaining the Decision of Anomalous Sound Detectors.” In: *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022.
- [141] John Martinsson, Martin Willbo, Aleksis Pirinen, Olof Mogren, and Maria Sandsten. “Few-Shot Bioacoustic Event Detection Using an Event-Length Adapted Ensemble of Prototypical Networks.” In: *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022.
- [142] Mark Mazumder, Colby R. Banbury, Josh Meyer, Pete Warden, and Vijay Janapa Reddi. “Few-Shot Keyword Spotting in Any Language.” In: *22nd Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA, 2021, pp. 4214–4218.
- [143] Donna Katzman McClish. “Analyzing a portion of the ROC curve.” In: *Medical decision making* 9.3 (1989), pp. 190–195.
- [144] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. “UMAP: Uniform Manifold Approximation and Projection.” In: *J. Open Source Softw.* 3.29 (2018), p. 861.

- [145] Li Meirong, Zhang Shaoying, Cheng Chuanxu, and Xu Wen. “Query-by-Example on-Device Keyword Spotting using Convolutional Recurrent Neural Network and Connectionist Temporal Classification.” In: *6th International Conference on Intelligent Computing and Signal Processing (ICSP)*. 2021, pp. 1291–1294.
- [146] Raghav Menon, Herman Kamper, John A. Quinn, and Thomas Niesler. “Fast ASR-free and Almost Zero-resource Keyword Spotting Using DTW and CNNs for Humanitarian Monitoring.” In: *19th Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA, 2018, pp. 2608–2612.
- [147] Raghav Menon, Armin Saeb, Hugh Cameron, William Kibira, John A. Quinn, and Thomas Niesler. “Radio-browsing for developmental monitoring in Uganda.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5795–5799.
- [148] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. “Metrics for polyphonic sound event detection.” In: *Applied Sciences* 6.6 (2016), p. 162.
- [149] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. “Acoustic Scene Classification in DCASE 2019 Challenge: Closed and Open Set Classification and Data Mismatch Setups.” In: *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. 2019, pp. 164–168.
- [150] Assaf Hurwitz Michaely, Xuedong Zhang, Gabor Simko, Carolina Parada, and Petar S. Aleksic. “Keyword spotting for Google assistant using contextual speech recognition.” In: *Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 272–278.
- [151] Saumitra Mishra, Bob L. Sturm, and Simon Dixon. “Local Interpretable Model-Agnostic Explanations for Music Content Analysis.” In: *18th International Society for Music Information Retrieval Conference (ISMIR)*. 2017, pp. 537–543.
- [152] Abdelrahman Mohamed et al. “Self-Supervised Speech Representation Learning: A Review.” In: *IEEE J. Sel. Top. Signal Process.* 16.6 (2022), pp. 1179–1210.
- [153] Veronica Morfi, Inês Nolasco, Vincent Lostanlen, Shubhr Singh, Ariana Strandburg-Peshkin, Lisa F. Gill, Hanna Pamula, David Benvent, and Dan Stowell. “Few-Shot Bioacoustic Event Detection: A New Task at the DCASE 2021 Challenge.” In: *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. 2021, pp. 145–149.
- [154] Kazuki Morita, Tomohiko Yano, and Khai Tran. *Anomalous Sound Detection Using CNN-Based Features By Self Supervised Learning*. Tech. rep. DCASE2021 Challenge, 2021.

- [155] Kazuki Morita, Tomohiko Yano, and Khai Tran. *Comparative Experiments on Spectrogram Representation for Anomalous Sound Detection*. Tech. rep. DCASE Challenge, 2022.
- [156] Mary M. Moya and Don R. Hush. “Network constraints and multi-objective optimization for one-class classification.” In: *Neural Networks 9.3* (1996), pp. 463–474.
- [157] Ami Moyal, Vered Aharonson, Ella Tetariy, and Michal Gishri. *Phonetic Search Methods for Large Speech Databases*. Springer Briefs in Electrical and Computer Engineering. Springer, 2013.
- [158] Robert Müller, Steffen Illium, Fabian Ritz, and Kyrill Schmid. “Analysis of Feature Representations for Anomalous Sound Detection.” In: *13th International Conference on Agents and Artificial Intelligence (ICAART)*. SCITEPRESS, 2021, pp. 97–106.
- [159] Robert Müller, Fabian Ritz, Steffen Illium, and Claudia Linnhoff-Popien. “Acoustic Anomaly Detection for Machine Sounds based on Image Transfer Learning.” In: *13th International Conference on Agents and Artificial Intelligence (ICAART)*. SCITEPRESS, 2021, pp. 49–56.
- [160] Shreesha Narasimha Murthy and Emmanuel Agu. “Deep Learning Anomaly Detection methods to passively detect COVID-19 from Audio.” In: *International Conference on Digital Health (ICDH)*. IEEE, 2021, pp. 114–121.
- [161] Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. “Do Deep Generative Models Know What They Don’t Know?” In: *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.
- [162] Javier Naranjo-Alcazar, Sergi Perez-Castanos, Pedro Zuccarello, Ana M. Torres, Jose J. Lopez, Francesc J. Ferri, and Maximo Cobos. “An open-set recognition and few-shot learning dataset for audio event classification in domestic environments.” In: *Pattern Recognition Letters* (2022).
- [163] Hiroki Narita and Akira Tamamori. *Unsupervised Anomalous Sound Detection Using Intermediate Representation of Trained Models and Metric Learning Based Variational Autoencoder*. Tech. rep. DCASE2021 Challenge, 2021.
- [164] Ismail Nejjar, Jean Meunier-Pion, Gaëtan Frusque, and Olga Fink. “DG-Mix: Domain Generalization for Anomalous Sound Detection Based on Self-Supervised Learning.” In: *7th Workshop on Detection and Classification of Acoustic Scenes and Events 2022 (DCASE)*. Tampere University, 2022.
- [165] Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino. “BYOL for Audio: Self-Supervised Learning for General-Purpose Audio Representation.” In: *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

- [166] Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino. “Composing General Audio Representation by Fusing Multilayer Features of a Pre-trained Model.” In: *30th European Signal Processing Conference (EUSIPCO)*. IEEE, 2022, pp. 200–204.
- [167] Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino. “BYOL for Audio: Exploring Pre-Trained General-Purpose Audio Representations.” In: *IEEE/ACM Trans. Audio Speech Lang. Process.* 31 (2023), pp. 137–151.
- [168] Tomoya Nishida, Kota Dohi, Takashi Endo, Masaaki Yamamoto, and Yohei Kawaguchi. “Anomalous Sound Detection Based on Machine Activity Detection.” In: *30th European Signal Processing Conference (EUSIPCO)*. IEEE, 2022, pp. 269–273.
- [169] Inês Nolasco et al. “Few-Shot Bioacoustic Event Detection at the DCASE 2022 Challenge.” In: *7th Workshop on Detection and Classification of Acoustic Scenes and Events 2022 (DCASE)*. 2022, pp. 136–140.
- [170] Stavros Ntalampiras and Ilyas Potamitis. “Acoustic detection of unknown bird species and individuals.” In: *CAAI Transactions on Intelligence Technology* 6.3 (2021), pp. 291–300.
- [171] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. “WaveNet: A Generative Model for Raw Audio.” In: *The 9th ISCA Speech Synthesis Workshop (SSW)*. ISCA, 2016, p. 125.
- [172] George Papamakarios, Eric T. Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. “Normalizing Flows for Probabilistic Modeling and Inference.” In: *J. Mach. Learn. Res.* 22 (2021), 57:1–57:64.
- [173] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition.” In: *20th Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA, 2019, pp. 2613–2617.
- [174] Jihwan Park and Sooyeon Yoo. “DCASE 2020 Task2: Anomalous Sound Detection using Relevant Spectral Feature and Focusing Techniques in the Unsupervised Learning Scenario.” In: *5th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. 2020, pp. 140–144.
- [175] Archit Parnami and Minwoo Lee. “Few-Shot Keyword Spotting With Prototypical Networks.” In: *7th International Conference on Machine Learning Technologies (ICMLT)*. ACM, 2022, pp. 277–283.
- [176] Fabian Pedregosa et al. “Scikit-learn: Machine Learning in Python.” In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

- [177] Pramuditha Perera and Vishal M. Patel. “Learning Deep Features for One-Class Classification.” In: *IEEE Transactions on Image Processing* 28.11 (2019), pp. 5450–5463.
- [178] Ricardo Falcón Pérez, Gordon Wichern, François G. Germain, and Jonathan Le Roux. “Location as Supervision for Weakly Supervised Multi-Channel Source Separation of Machine Sounds.” In: *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2023.
- [179] Sergi Perez-Castanos, Javier Naranjo-Alcazar, Pedro Zuccarello, and Maximo Cobos. “Anomalous Sound Detection using Unsupervised and Semi-Supervised Autoencoders and Gammatone Audio Representation.” In: *5th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. 2020, pp. 145–149.
- [180] François Petitjean, Alain Ketterlin, and Pierre Gançarski. “A global averaging method for dynamic time warping, with applications to clustering.” In: *Pattern recognition* 44.3 (2011), pp. 678–693.
- [181] Vitali Petsiuk, Abir Das, and Kate Saenko. “RISE: Randomized Input Sampling for Explanation of Black-box Models.” In: *British Machine Vision Conference (BMVC)*. BMVA Press, 2018, p. 151.
- [182] Lam Pham, Anahid Jalali, Olivia Dinica, and Alexander Schindler. *DCASE Challenge 2021: Unsupervised Anomalous Sound Detection of Machinery with LeNet Architecture*. Tech. rep. DCASE2021 Challenge, 2021.
- [183] Paul Primus. *Reframing Unsupervised Machine Condition Monitoring as a Supervised Classification Task with Outlier-Exposed Classifiers*. Tech. rep. DCASE2020 Challenge, 2020.
- [184] Paul Primus, Verena Haunschmid, Patrick Praher, and Gerhard Widmer. “Anomalous Sound Detection as a Simple Binary Classification Problem with Careful Selection of Proxy Outlier Examples.” In: *5th Workshop on Detection and Classification of Acoustic Scenes and Events 2020 (DCASE)*. 2020, pp. 170–174.
- [185] Simon J.D. Prince and James H. Elder. “Probabilistic linear discriminant analysis for inferences about identity.” In: *11th International Conference on Computer Vision (ICCV)*. IEEE. 2007.
- [186] Harsh Purohit, Takashi Endo, Masaaki Yamamoto, and Yohei Kawaguchi. “Hierarchical Conditional Variational Autoencoder Based Acoustic Anomaly Detection.” In: *30th European Signal Processing Conference (EU-SIPCO)*. IEEE, 2022, pp. 274–278.
- [187] Harsh Purohit, Ryo Tanabe, Takashi Endo, Kaori Suefusa, Yuki Nikaido, and Yohei Kawaguchi. “Deep Autoencoding GMM-Based Unsupervised Anomaly Detection in Acoustic Signals and its Hyper-Parameter Optimization.” In: *5th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. 2020, pp. 175–179.

- [188] Harsh Purohit, Ryo Tanabe, Takeshi Ichige, Takashi Endo, Yuki Nikaido, Kaori Suefusa, and Yohei Kawaguchi. “MIMII Dataset: Sound Dataset for Malfunctioning Industrial Machine Investigation and Inspection.” In: *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. New York University, 2019, pp. 209–213.
- [189] Clemens Reimann, Peter Filzmoser, and Robert G. Garrett. “Background and threshold: critical comparison of methods of determination.” In: *Science of the total environment* 346.1-3 (2005), pp. 1–16.
- [190] Tal Reiss, Niv Cohen, Liron Bergman, and Yedid Hoshen. “PANDA: Adapting Pretrained Features for Anomaly Detection and Segmentation.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE, 2021, pp. 2806–2814.
- [191] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. ““Why Should I Trust You?”: Explaining the Predictions of Any Classifier.” In: *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1135–1144.
- [192] Aaron E. Rosenberg, Chin-Hui Lee, and Frank K. Soong. “Cepstral channel normalization techniques for HMM-based speaker verification.” In: *The 3rd International Conference on Spoken Language Processing (ICSLP)*. ISCA, 1994.
- [193] Bernard Rosner. “Percentage Points for a Generalized ESD Many-Outlier Procedure.” In: *Technometrics* 25.2 (1983), pp. 165–172.
- [194] Peter J. Rousseeuw and Christophe Croux. “Alternatives to the median absolute deviation.” In: *Journal of the American Statistical association* 88.424 (1993), pp. 1273–1283.
- [195] Peter J. Rousseeuw and Bert C. Van Zomeren. “Unmasking multivariate outliers and leverage points.” In: *Journal of the American Statistical association* 85.411 (1990), pp. 633–639.
- [196] Lukas Ruff. “Deep one-class learning: a deep learning approach to anomaly detection.” PhD thesis. Technical University of Berlin, Germany, 2021.
- [197] Lukas Ruff, Nico Görnitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Robert A. Vandermeulen, Alexander Binder, Emmanuel Müller, and Marius Kloft. “Deep One-Class Classification.” In: *35th International Conference on Machine Learning (ICML)*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 4390–4399.
- [198] Yuya Sakamoto and Naoya Miyamoto. *Combine Mahalanobis Distance, Interpolation Auto Encoder and Classification Approach for Anomaly Detection*. Tech. rep. DCASE2021 Challenge, 2021.

- [199] Johan Schalkwyk, Doug Beeferman, Françoise Beaufays, Bill Byrne, Ciprian Chelba, Mike Cohen, Maryam Kamvar, and Brian Strope. ““Your word is my command”: Google search by voice: A case study.” In: *Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics* (2010), pp. 61–90.
- [200] Walter J. Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrence E. Boult. “Toward Open Set Recognition.” In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.7 (2013), pp. 1757–1772.
- [201] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. “Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery.” In: *25th International Conference on Information Processing in Medical Imaging (IPMI)*. Vol. 10265. Lecture Notes in Computer Science. Springer, 2017, pp. 146–157.
- [202] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alexander J. Smola, and Robert C. Williamson. “Estimating the Support of a High-Dimensional Distribution.” In: *Neural Comput.* 13.7 (2001), pp. 1443–1471.
- [203] Kanta Shimonishi, Kota Dohi, and Yohei Kawaguchi. “Anomalous Sound Detection Based on Sound Separation.” In: *24th Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA, 2023, pp. 2733–2737.
- [204] Stepan Shishkin, Danilo Hollosi, Simon Doclo, and Stefan Goetze. “Active Learning for Sound Event Classification using Monte-Carlo Dropout and PANN Embeddings.” In: *6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. 2021, pp. 150–154.
- [205] Suwon Shon, Najim Dehak, Douglas A. Reynolds, and James R. Glass. “MCE 2018: The 1st Multi-Target Speaker Detection and Identification Challenge Evaluation.” In: *20th Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA, 2019, pp. 356–360.
- [206] Robert H. Shumway and David S. Stoffer. *Time series analysis and its applications*. Vol. 3. Springer, 2000.
- [207] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” In: *3rd International Conference on Learning Representations (ICLR)*. 2015.
- [208] Aleksandr Sizov, Kong Aik Lee, and Tomi Kinnunen. “Unifying Probabilistic Linear Discriminant Analysis Variants in Biometric Authentication.” In: *Proc. S+SSPR*. Software available at <https://sites.google.com/site/fastplda/>. Springer. 2014, pp. 464–475.

- [209] Jake Snell, Kevin Swersky, and Richard S. Zemel. “Prototypical Networks for Few-shot Learning.” In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS)*. 2017, pp. 4077–4087.
- [210] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. “X-Vectors: Robust DNN Embeddings for Speaker Recognition.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.
- [211] Stanley Smith Stevens, John Volkman, and Edwin Broomell Newman. “A scale for the measurement of the psychological magnitude pitch.” In: *The Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190.
- [212] Kaori Suefusa, Tomoya Nishida, Harsh Purohit, Ryo Tanabe, Takashi Endo, and Yohei Kawaguchi. “Anomalous Sound Detection Based on Interpolation Deep Neural Network.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 271–275.
- [213] Cecilia Summers and Michael J. Dinneen. “Improved Mixed-Example Data Augmentation.” In: *Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1262–1270.
- [214] Yanmin Sun, Andrew K. C. Wong, and Mohamed S. Kamel. “Classification of Imbalanced Data: a Review.” In: *Int. J. Pattern Recognit. Artif. Intell.* 23.4 (2009), pp. 687–719.
- [215] Jiantong Tian, Hejing Zhang, Qiaoxi Zhu, Feiyang Xiao, Haohe Liu, Xinhao Mei, Youde Liu, Wenwu Wang, and Jian Guan. *First-shot Anomalous Sound Detection with GMM Clustering and Finetuned Attribute Classification using Audio Pretrained Model*. Tech. rep. DCASE2023 Challenge, 2023.
- [216] Giuseppe Valenzise, Luigi Gerosa, Marco Tagliasacchi, Fabio Antonacci, and Augusto Sarti. “Scream and gunshot detection and localization for audio-surveillance systems.” In: *Fourth International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2007, pp. 21–26.
- [217] Satvik Venkatesh, Gordon Wichern, Aswin Subramanian, and Jonathan Le Roux. *Disentangled surrogate task learning for improved domain generalization in unsupervised anomalous sound detection*. Tech. rep. DCASE Challenge, 2022.
- [218] Satvik Venkatesh, Gordon Wichern, Aswin Shanmugam Subramanian, and Jonathan Le Roux. “Improved Domain Generalization via Disentangled Multi-Task Learning in Unsupervised Anomalous Sound Detection.” In: *7th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*. Tampere University, 2022, pp. 196–200.

- [219] Sergey Verbitskiy, Milana Shkhanukova, and Viacheslav Vyshegorodtsev. *Unsupervised Anomalous Sound Detection Using Multiple Time-Frequency Representations*. Tech. rep. DCASE Challenge, 2022.
- [220] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. “Manifold Mixup: Better Representations by Interpolating Hidden States.” In: *36th International Conference on Machine Learning (ICML)*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 6438–6447.
- [221] Phongtharin Vinayavekhin, Tadanobu Inoue, Shu Morikuni, Shiqiang Wang, Tuan Hoang Trong, David Wood, Michiaki Tatsubori, and Ryuki Tachibana. *Detection of Anomalous Sounds for Machine Condition Monitoring using Classification Confidence*. Tech. rep. DCASE2020 Challenge, 2020.
- [222] Dirk Von Zeddelmann, Frank Kurth, and Meinard Müller. “Perceptual audio features for unsupervised key-phrase detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2010, pp. 257–260.
- [223] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. “Additive Margin Softmax for Face Verification.” In: *IEEE Signal Processing Letters* 25.7 (2018), pp. 926–930.
- [224] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. “CosFace: Large Margin Cosine Loss for Deep Face Recognition.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 5265–5274.
- [225] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, and Tao Qin. “Generalizing to Unseen Domains: A Survey on Domain Generalization.” In: *Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*. ijcai.org, 2021, pp. 4627–4635.
- [226] Lei Wang et al. *First-Shot Unsupervised Anomalous Sound Detection Using Attribute Classification and Conditional Autoencoder*. Tech. rep. DCASE2023 Challenge, 2023.
- [227] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. “Generalizing from a Few Examples: A Survey on Few-shot Learning.” In: *ACM Comput. Surv.* 53.3 (2021), 63:1–63:34.
- [228] Yu Wang, Nicholas J. Bryan, Mark Cartwright, Juan Pablo Bello, and Justin Salamon. “Few-Shot Continual Learning for Audio Classification.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 321–325.

- [229] Yu Wang, Mark Cartwright, and Juan Pablo Bello. “Active Few-Shot Learning for Sound Event Detection.” In: *23rd Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA, 2022, pp. 1551–1555.
- [230] Yu Wang, Justin Salamon, Nicholas J. Bryan, and Juan Pablo Bello. “Few-Shot Sound Event Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 81–85.
- [231] Pete Warden. “Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition.” In: *CoRR* abs/1804.03209 (2018).
- [232] Yuming Wei, Jian Guan, Haiyan Lan, and Wenwu Wang. *Anomalous Sound Detection System with Self-challenge and Metric Evaluation for DCASE2022 Challenge Task 2*. Tech. rep. DCASE Challenge, 2022.
- [233] Kilian Q. Weinberger and Lawrence K. Saul. “Distance metric learning for large margin nearest neighbor classification.” In: *Journal of machine learning research* 10.2 (2009).
- [234] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. “A discriminative feature learning approach for deep face recognition.” In: *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 499–515.
- [235] Gordon Wichern, Ankush Chakrabarty, Zhong-Qiu Wang, and Jonathan Le Roux. “Anomalous Sound Detection Using Attentive Neural Processes.” In: *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2021, pp. 186–190.
- [236] Kevin Wilkinghoff. “General-Purpose Audio Tagging by Ensembling Convolutional Neural Networks based on Multiple Features.” In: *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. Tampere University of Technology, 2018, pp. 44–48.
- [237] Kevin Wilkinghoff. “On Open-Set Classification with L3-Net Embeddings for Machine Listening Applications.” In: *28th European Signal Processing Conference (EUSIPCO)*. IEEE, 2020, pp. 800–804.
- [238] Kevin Wilkinghoff. “On Open-Set Speaker Identification with I-Vectors.” In: *Odyssey - The Speaker and Language Recognition Workshop (Odyssey)*. ISCA, 2020, pp. 408–414.
- [239] Kevin Wilkinghoff. “Using Look, Listen, and Learn Embeddings for Detecting Anomalous Sounds in Machine Condition Monitoring.” In: *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. 2020, pp. 215–219.
- [240] Kevin Wilkinghoff. “Combining Multiple Distributions based on Sub-Cluster AdaCos for Anomalous Sound Detection under Domain Shifted Conditions.” In: *Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*. 2021.

- [241] Kevin Wilkinghoff. “Sub-Cluster AdaCos: Learning Representations for Anomalous Sound Detection.” In: *International Joint Conference on Neural Networks*. IEEE, 2021. DOI: [10.1109/IJCNN52387.2021.9534290](https://doi.org/10.1109/IJCNN52387.2021.9534290).
- [242] Kevin Wilkinghoff. “Sub-Cluster AdaCos: Learning Representations for Anomalous Sound Detection.” In: *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.
- [243] Kevin Wilkinghoff. *An outlier exposed anomalous sound detection system for domain generalization in machine condition monitoring*. Tech. rep. DCASE Challenge, 2022.
- [244] Kevin Wilkinghoff. “Design Choices for Learning Embeddings from Auxiliary Tasks for Domain Generalization in Anomalous Sound Detection.” In: *International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2023. DOI: [10.1109/ICASSP49357.2023.10097176](https://doi.org/10.1109/ICASSP49357.2023.10097176).
- [245] Kevin Wilkinghoff. “Design Choices for Learning Embeddings from Auxiliary Tasks for Domain Generalization in Anomalous Sound Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.
- [246] Kevin Wilkinghoff. *Fraunhofer FKIE submission for Task 2: First-Shot Un-supervised Anomalous Sound Detection for Machine Condition Monitoring*. Tech. rep. DCASE2023 Challenge, 2023.
- [247] Kevin Wilkinghoff. “AdaProj: Adaptively Scaled Angular Margin Subspace Projections for Anomalous Sound Detection with Auxiliary Classification Tasks.” Submitted to 9th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE), arXiv:2403.14179. 2024. DOI: [10.48550/arXiv.2403.14179](https://doi.org/10.48550/arXiv.2403.14179).
- [248] Kevin Wilkinghoff. “AdaProj: Adaptively Scaled Angular Margin Subspace Projections for Anomalous Sound Detection with Auxiliary Classification Tasks.” Submitted to 9th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE), arXiv:2403.14179. 2024.
- [249] Kevin Wilkinghoff. “Self-Supervised Learning for Anomalous Sound Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 276–280. DOI: [10.1109/ICASSP48485.2024.10447156](https://doi.org/10.1109/ICASSP48485.2024.10447156).
- [250] Kevin Wilkinghoff. “Self-Supervised Learning for Anomalous Sound Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 276–280.
- [251] Kevin Wilkinghoff and Alessia Cornaggia-Urrigshardt. “On choosing decision thresholds for anomalous sound detection in machine condition monitoring.” In: *24th International Congress on Acoustics*. The Acoustical Society of Korea, 2022.

- [252] Kevin Wilkinghoff and Alessia Cornaggia-Urrigshardt. “On choosing decision thresholds for anomalous sound detection in machine condition monitoring.” In: *24th International Congress on Acoustics (ICA)*. The Acoustical Society of Korea, 2022.
- [253] Kevin Wilkinghoff and Alessia Cornaggia-Urrigshardt. “TACos: Learning Temporally Structured Embeddings for Few-Shot Keyword Spotting with Dynamic Time Warping.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 9941–9945. DOI: [10.1109/ICASSP48485.2024.10445814](https://doi.org/10.1109/ICASSP48485.2024.10445814).
- [254] Kevin Wilkinghoff and Alessia Cornaggia-Urrigshardt. “TACos: Learning Temporally Structured Embeddings for Few-Shot Keyword Spotting with Dynamic Time Warping.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 9941–9945.
- [255] Kevin Wilkinghoff, Alessia Cornaggia-Urrigshardt, and Fahrettin Gökgöz. “Two-Dimensional Embeddings for Low-Resource Keyword Spotting Based on Dynamic Time Warping.” In: *14th ITG Conference on Speech Communication*. VDE-Verlag, 2021, pp. 9–13.
- [256] Kevin Wilkinghoff, Alessia Cornaggia-Urrigshardt, and Fahrettin Gökgöz. “Two-Dimensional Embeddings for Low-Resource Keyword Spotting Based on Dynamic Time Warping.” In: *14th ITG Conference on Speech Communication (ITG Speech)*. VDE-Verlag, 2021, pp. 9–13.
- [257] Kevin Wilkinghoff and Fabian Fritz. “On Using Pre-Trained Embeddings for Detecting Anomalous Sounds with Limited Training Data.” In: *31st European Signal Processing Conference*. IEEE, 2023, pp. 186–190. DOI: [10.23919/EUSIPCO58844.2023.10290003](https://doi.org/10.23919/EUSIPCO58844.2023.10290003).
- [258] Kevin Wilkinghoff and Fabian Fritz. “On Using Pre-Trained Embeddings for Detecting Anomalous Sounds with Limited Training Data.” In: *31st European Signal Processing Conference (EUSIPCO)*. IEEE, 2023, pp. 186–190.
- [259] Kevin Wilkinghoff and Keisuke Imoto. “F1-EV Score: Measuring the Likelihood of Estimating a Good Decision Threshold for Semi-Supervised Anomaly Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 256–260. DOI: [10.1109/ICASSP48485.2024.10446011](https://doi.org/10.1109/ICASSP48485.2024.10446011).
- [260] Kevin Wilkinghoff and Keisuke Imoto. “F1-EV Score: Measuring the Likelihood of Estimating a Good Decision Threshold for Semi-Supervised Anomaly Detection.” In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 256–260.

- [261] Kevin Wilkinghoff and Frank Kurth. “Why do Angular Margin Losses work well for Semi-Supervised Anomalous Sound Detection?” In: *IEEE/ACM Transactions on Audio, Speech and Language Processing* 32 (2024), pp. 608–622. DOI: [10.1109/TASLP.2023.3337153](https://doi.org/10.1109/TASLP.2023.3337153).
- [262] Kevin Wilkinghoff and Frank Kurth. “Why do Angular Margin Losses work well for Semi-Supervised Anomalous Sound Detection?” In: *IEEE/ACM Trans. Audio Speech Lang. Process.* 32 (2024), pp. 608–622.
- [263] Huang Xie and Tuomas Virtanen. “Zero-Shot Audio Classification Via Semantic Embeddings.” In: *IEEE/ACM Trans. Audio Speech Lang. Process.* 29 (2021), pp. 1233–1242.
- [264] Jia Yafei, Bai Jisheng, and Huang Siwei. *Unsupervised Abnormal Sound Detection Based on Machine Condition Mixup*. Tech. rep. DCASE2023 Challenge, 2023.
- [265] Peng Yan, Ahmed Abdulkadir, Paul-Philipp Luley, Matthias Rosenthal, Gerrit A. Schatte, Benjamin F. Grewe, and Thilo Stadelmann. “A Comprehensive Survey of Deep Transfer Learning for Anomaly Detection in Industrial Time Series: Methods, Applications, and Directions.” In: *IEEE Access* 12 (2024), pp. 3768–3789.
- [266] Hanfang Yang, Kun Lu, Xiang Lyu, and Feifang Hu. “Two-way partial AUC and its properties.” In: *Statistical methods in medical research* 28.1 (2019), pp. 184–195.
- [267] Jiawei Yang, Susanto Rahardja, and Pasi Fränti. “Outlier detection: how to threshold outlier scores?” In: *International Conference on Artificial Intelligence, Information Processing and Cloud Computing (AIIPCC)*. ACM, 2019, 37:1–37:6.
- [268] Qien Yu, Muthu Subash Kavitha, and Takio Kurita. “Autoencoder framework based on orthogonal projection constraints improves anomalies detection.” In: *Neurocomputing* 450 (2021), pp. 372–388.
- [269] Ying Zeng, Hongqing Liu, Lihua Xu, Yi Zhou, and Lu Gan. *Robust Anomaly Sound Detection Framework for Machine Condition Monitoring*. Tech. rep. DCASE Challenge, 2022.
- [270] Chenxu Zhang, Yao Yao, Rui Qiu, Shengchen Li, and Xi Shao. *Unsupervised Anomalous Sound Detection Using Denoising-Detection System Under Domain Shifted Conditions*. Tech. rep. DCASE2021 Challenge, 2021.
- [271] Hanlin Zhang, Yi-Fan Zhang, Weiyang Liu, Adrian Weller, Bernhard Schölkopf, and Eric P. Xing. “Towards Principled Disentanglement for Domain Generalization.” In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022, pp. 8014–8024.

- [272] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. “Mixup: Beyond empirical risk minimization.” In: *6th International Conference on Learning Representations (ICLR)*. Openreview.net, 2018.
- [273] Minghu Zhang, Xin Li, and Lili Wang. “An Adaptive Outlier Detection and Processing Approach Towards Time Series Sensor Data.” In: *IEEE Access* 7 (2019), pp. 175192–175212.
- [274] Xiao Zhang, Rui Zhao, Yu Qiao, Xiaogang Wang, and Hongsheng Li. “AdaCos: Adaptively Scaling Cosine Logits for Effectively Learning Deep Face Representations.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 10823–10832.
- [275] Shuyang Zhao, Toni Heittola, and Tuomas Virtanen. “Active Learning for Sound Event Detection.” In: *IEEE/ACM Trans. Audio Speech Lang. Process.* 28 (2020), pp. 2895–2905.
- [276] Qiping Zhou. *ArcFace Based Sound Mobilenets for DCASE 2020 Task 2*. Tech. rep. DCASE2020 Challenge, 2020.
- [277] Yifan Zhou and Yanhua Long. *Attribute Classifier with Imbalance Compensation for Anomalous Sound Detection*. Tech. rep. DCASE2023 Challenge, 2023.
- [278] Manli Zhu and Aleix M. Martinez. “Subclass discriminant analysis.” In: *IEEE Trans. Pattern Anal. Mach. Intell.* 28.8 (2006), pp. 1274–1286.
- [279] Christian Zieger, Alessio Brutti, and Piergiorgio Svaizer. “Acoustic Based Surveillance System for Intrusion Detection.” In: *6th International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2009, pp. 314–319.