

Utilization of Reconstructive Representation Learning for Robust Classification

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

von
Max Lübbering
aus
Borken

Bonn, 2023

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

1. Gutachter: Prof. Dr. Christian Bauckhage
2. Gutachter: Prof. Dr. Stefan Wrobel

Tag der Promotion: 27. Juni 2023
Erscheinungsjahr: 2023

Acknowledgements

As with any thesis, the success of this PhD thesis heavily depended on the support, feedback, encouragement, and understanding of my advisors, supervisors, colleagues, family, and friends. Throughout the course of my PhD program, I was fortunate enough to have been supported in every way unsolicitedly.

Firstly, I would like to thank my advisor Dr. Rafet Sifa. From the first paper experiments to polishing this thesis, Rafet has guided me in the right directions, challenged my perspectives, and taught me how to ask the right research questions. His technical expertise, motivation and critical thinking have greatly benefitted this thesis and my personal development.

Secondly, I would like to thank my supervisors Prof. Dr. Christian Bauckhage and Prof. Dr. Stefan Wrobel, for accepting me as their PhD student, trusting me to conduct my own research, and finally, grading my PhD work.

Thirdly, my colleagues Rajkumar Ramamurthy and Michael Gebauer deserve credit for their extensive contributions to my PhD research papers. The countless hours of research discussions and experiment analysis in the evenings and weekends have turned into exciting research results and long-lasting friendships. Additionally, special thanks go out to my current or former colleagues at Fraunhofer IAIS, Rhys Agombar, Maren Pielka, Helen Schneider, Priya Priya, Thiago Bell, Lars Hillebrand, Robin Stenzel, Bilge Ulusay, Daniel Uedelhoven, David Biesner, Ulrich Nütten, Cengiz Henk, Lorenz Wickert, Ben Wulff, Osama Soliman and many others who always kept me motivated, introduced me to their field of work and made working at Fraunhofer IAIS such an amazing experience. In particular, I thank Rhys and Lorenz for their critical feedback on this thesis.

Finally, I would like to thank my family and friends. Throughout the PhD, my girlfriend Sofia never stopped providing me with the emotional support I needed in times of struggle. Her non-judgmental, caring and loving personality gave me the backing and self-confidence I needed, preventing me from losing myself in this thesis. I thank my mom for her love, understanding, and always being there for me, and my dad for his love and vision. I thank my brother and friends for being in my life and making it so exciting beyond my profession.

Thanks.

Contents

1	Introduction	9
1.1	Optimization of Deep Neural Networks	12
1.2	Introduction to Outlier Detection	16
1.2.1	Evaluating Outlier Detection and Open-set Recognition Systems	17
1.2.2	Traditional Outlier Detection Methods	18
1.2.3	Deep (Supervised) Outlier Detection	19
1.2.4	Relationship Between Outlier Detection, Imbalanced Classification, Open Set Recognition, One-class Classification, Novelty Detection, and One-vs-rest Classification	20
1.3	Autoencoders From a Manifold Learning Perspective	20
1.4	Robust multi-class classification	22
1.5	Outline of This Work	22
2	Design of the Experiment Environment	25
2.1	Datastack: Unification of Heterogeneous Machine Learning Dataset Interfaces	26
2.1.1	Design Choices	28
2.1.2	Implementation	28
2.1.3	Use case: Dataset Processing Pipeline for Outlier Detection	30
2.1.4	Conclusion and Outlook	30
2.2	MLgym: Architectural Proposal for Reproducible, Standardized Deep Learning Research	31
2.2.1	Related Work	33
2.2.2	Architectural Overview	33
2.2.3	System Design	37
2.2.4	Representative Research Use Case	38
2.2.5	Quo vadis?	40
2.2.6	Conclusion and Outlook	41
2.3	Conclusion	41
3	Supervised Outlier Detection with Deep Neural Networks	43
3.1	Introduction	44
3.2	Autoencoders for Outlier and Novelty Detection	45
3.3	Related Work	46
3.4	Evaluation	46
3.5	From Imbalanced Classification to Supervised Outlier Detection Problems	48
3.5.1	Adversarially Trained Autoencoders	48
3.5.2	Experiments and Results	52

3.5.3	Conclusion	54
3.6	Supervised Autoencoder Variants for End-to-end Anomaly Detection	54
3.6.1	Supervised Autoencoders	55
3.6.2	Adversarial Supervised Autoencoders	55
3.6.3	Experiments and Results	57
3.6.4	Conclusion	59
3.7	Summary and Outlook	60
4	Open Set Recognition	63
4.1	Introduction	64
4.2	Related Work	67
4.3	Decoupling Autoencoders	68
4.4	Classification Concern Conflicting with Robustness	71
4.5	Achieving Bounded Open Set Recognition with Autoencoders	72
4.6	Towards Adversarial Robustness and Local Stability	74
4.7	Experiments and Results	75
4.7.1	Selected Baselines	75
4.7.2	Evaluation Approach	76
4.7.3	Datasets	77
4.7.4	Results	78
4.8	Conclusion	86
5	From Open Set Recognition Towards Robust Multi-class Classification	89
5.1	Introduction	90
5.2	Related Work	92
5.3	Informer	92
5.4	Evaluation Approach	95
5.5	Results	97
5.6	Conclusion	100
6	Applications	101
6.1	Toxicity Detection in Online Comments with Limited Data: A Comparative Analysis	102
6.1.1	Toxicity Detection Dataset	103
6.1.2	Experiments	104
6.1.3	Results	104
6.1.4	Challenges of Encoding Toxicities	106
6.1.5	Conclusion	106
6.2	Deployment Case Study: Automatic Indexing of Financial Documents via Information Extraction	107
6.2.1	Extraction Service	109
6.2.2	Experiments and Results	112
6.2.3	Conclusion and Outlook	115
7	Conclusion	117
7.1	A General Summary	117

7.2 Outlook	119
A Appendix	121
A.1 From Open Set Recognition Towards Robust Multi-class Classification	121
A.1.1 Derivation of Informer Component Interdependence Induced by Softmax . .	121
A.1.2 Derivation of Uncertainty Computation Within Uncertainty Estimation Module	122
A.2 MLgym exemplary pipeline configuration	123
Bibliography	129
List of Figures	149
List of Tables	155
Publications	159

Abstract

Deep neural networks (DNNs) are generally trained via empirical risk minimization (ERM) on classification tasks. While this has led to impressive results in scientific benchmarks, as well as, industrial applications, it has been also shown that DNNs tend to give wrong predictions with elevated confidence on out-of-distribution data. In the past, various AI accidents have been associated with these robustness deficiencies of DNNs, making the development of safer DNN architectures inevitable.

To this end, we examine this issue from a theoretical, optimizational point of view and empirically verify the deficiency across various benchmarks. As a potential, multistep solution, we turn towards outlier detection methods in the first step, as such methods aim to capture out-of-distribution data, i.e., data that cannot be explained by the data generating process of normal data. In particular, we utilize reconstructive representation learning, i.e., autoencoders, to learn a representation of normality and leverage the reconstruction error as an outlieriness signal to filter outliers. We find that the integration of outlier data into the training process, as opposed to previous works (e.g., *one-class autoencoders*), benefits the model robustness significantly, and propose the novel architecture *adversarially trained autoencoder* (ATA), which includes this insight by actively maximizing/minimizing the reconstruction error of outliers/inliers, respectively.

In the second step, we consider the related problem of open-set recognition (OSR), which aims to filter a fixed set of inlier classes from all the possibly existing rest classes including out-of-distribution data. We show that our supervised outlier detection method ATA can solve this generalized one-vs-rest classification task, without expressing the robustness deficiencies of DNNs optimized via ERM. To actively reduce the open-space risk, a principal robustness criterion in OSR, we extend ATA towards our *decoupled autoencoder* (DAE) architecture, which learns a tighter hull around the inlier data and provides probability scores on the inlierness of a sample, in contrast to ATA. To support our empirical evidence, we prove the existence of an upper bound on the open-space risk for ATA and DAE.

In the final step, we perform multi-class classification on the inlier classes in the OSR setting, which resembles the multi-class classification of real-world deployments due to the out-of-distribution exposure. To this end, we compose an ensemble of DAEs, each learning a different one-vs-rest relationship on the inlier classes, and demonstrate the robustness benefits and its capability to separate between aleatoric and epistemic uncertainty. All three properties together are unmatched by any other DNN architecture.

Finally, the applicability to real-world settings is displayed on the use case of toxicity detection in online communication and the deployment case study of a large-scale information extraction system for financial data.

Preface

With the rise in GPU and CPU performance and data abundance in the past decades, deep neural networks (DNNs) and other machine learning (ML) methods underwent a quantum leap in innovation and still continuously advance into our daily lives when we think of, e.g., medical diagnosis systems, autonomous driving or stock trading. Despite their success in natural language processing (NLP), computer vision, and speech recognition, traditional DNNs are not the panacea to all ML problems. When exposed to unseen data, DNNs tend to give wrong predictions with elevated confidence, potentially resulting in devastating consequences when carelessly deployed in safety-critical systems. This raises the central question of this thesis:

How can we teach neural networks awareness for the unknown?

Alternatively, putting it in the colloquial words of this famous quotation, which ironically has often been accredited to Mark Twain without any substantial evidence of its true origins:

It ain't what you don't know that gets you into trouble. It's what you know for sure that just ain't so.

–Anonymous

Humans have a rich uncertainty model of the perceived environment that enables us to reason based on different uncertainty concepts. We can be epistemically uncertain about a prediction due to lack of important information, or we can be aleatorically uncertain due to environment-inherent stochastic processes (e.g., uncertain about the result of a coin flip). Distinguishing these types of uncertainties is crucial in many aspects of our lives. For instance, physicians always apply these concepts intuitively when they examine a patient.

During training, neural network classifiers learn to perform classification between a set of classes within a problem domain by minimizing the empirical risk within the data. These models generally provide high accuracy and reasonable aleatoric uncertainty estimates within the problem domain. However, they fail to capture epistemic uncertainty, as it is not reflected within the training process. Consequently, when we zoom out of the problem domain, DNNs do not realize when they do not know something.

We try to solve these deficits of (multi-class) classification in three steps. In the first step, we turn towards outlier detection methods which aim to detect points that deviate so significantly from the normal data that they cannot be explained by the inlier generating process. Thus, learning a representation for the known implicitly allows for taming the unknown. We follow this idea by developing supervised outlier detection methods based on reconstructive representation learning, i.e., autoencoders, that accurately predict outliers by learning a meaningful representation for inliers and utilizing the reconstruction error as an outlierness signal. As indicated by our results, incorporating outliers into the training process significantly improves the robustness of outlier detectors.

In the second step, we reframe the supervised outlier detection problem as a one-vs-rest (OVR) classification problem, which aims to distinguish a set of inlier classes from all rest classes within the problem domain. Extending the set of rest classes to all possibly existing rest classes and outliers, the OVR setting can be seen as an open-set recognition (OSR) problem that cannot be addressed with plain empirical risk minimization. We find that our supervised outlier detection methods can effectively separate between inlier classes and rest classes, and thus, are capable of rejecting the unknown.

In the final step, we regard a multi-class classification problem with C classes as C OVR classification problems. For each of the C inlier classes, we learn a robust OVR model based on our previously developed method. Thus, the robustness benefits extend to the multi-class classification problem. Furthermore, we show that our models capture both aleatoric and epistemic uncertainty and can distinguish the two, making them highly attractive for deployment in safety-critical environments.

This thesis is structured in six parts. After this introductory chapter, we present two open-source frameworks in Ch. 2 which provide the infrastructure for our experimental setup. The frameworks jointly streamline the training of DNNs and reduce the complexity of robustness / uncertainty estimation evaluation. Furthermore, they are problem-agnostic and can be applied to perform fast-paced, reproducible, and scalable deep learning (DL) research.

In Ch. 3, we show that unsupervised and semi-supervised outlier detection methods insufficiently separate inliers from outliers when the outliers are overly correlated with the inliers. As a solution, we propose the *adversarially trained autoencoder* (ATA) architecture, which actively maximizes the reconstruction error of outliers. Our results conclusively show that ATA outperforms each outlier detection and classification method on the imbalanced classification, outlier detection, and novelty detection tasks. We further investigate if ATA combined with the multi-task learning algorithm *supervised autoencoder* (SAE) [1] yields better latent representation for robust classification. While SAE generally outperforms the baselines, we find indications that adversarially learned representations are especially beneficial on novelty detection.

In Ch. 4, we formalize the concept of awareness of the unknown within the OSR framework. To this end, we propose the novel architecture *decoupling autoencoder* (DAE), which has a proven upper bound on the open space risk, a principal robustness criterion, and minimizes open space risk via a dedicated training routine. Our method is benchmarked with three different scenarios, each isolating different aspects of OSR, namely, plain classification, outlier detection, and dataset shift. The results conclusively show that DAE achieves robust performance across all three tasks. This level of cross-task robustness was not observed in any of the baseline methods.

Ch. 5 deals with robust multi-class classification. We utilize the robustness insights of the previous two chapters and integrate reconstructive representation learning for multi-class classification into our proposed Informer architecture. Our results show that Informers can separate epistemic and aleatoric uncertainty, a crucial advantage in safety-critical systems.

Finally, Ch. 6 presents a larger-scale application of our methods on toxicity detection in online comments and showcases a deployment case study of information extraction from financial documents. The thesis closes with a conclusion in Ch. 7.

This thesis is based on the following chronologically ordered publications published between 2020 and 2022:

- [1] M. Lübbering, R. Ramamurthy, M. Gebauer, T. Bell, R. Sifa and C. Bauckhage, “From Imbalanced Classification to Supervised Outlier Detection Problems: Adversarially

-
- Trained Auto Encoders”, *International Conference on Artificial Neural Networks*, Springer, 2020.
- [2] M. Lübbering, M. Gebauer, R. Ramamurthy, R. Sifa and C. Bauckhage, “Supervised Autoencoder Variants for End to End Anomaly Detection”, *Pattern Recognition. ICPR International Workshops and Challenges*, 2021.
- [3] M. Lübbering, M. Gebauer, R. Ramamurthy, M. Pielka, C. Bauckhage and R. Sifa, “Utilizing Representation Learning for Robust Text Classification Under Datasetshift”, *Proceedings of the Conference "Lernen, Wissen, Daten, Analysen"*, 2021.
- [4] M. Lübbering, M. Pielka, K. Das, M. Gebauer, R. Ramamurthy, C. Bauckhage and R. Sifa, “Toxicity Detection in Online Comments with Limited Data: A Comparative Analysis.”, *ESANN*, 2021.
- [5] M. Lübbering, M. Gebauer, R. Ramamurthy, C. Bauckhage and R. Sifa, “Decoupling Autoencoders for Robust One-vs-Rest Classification”, *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, 2021 1.
- [6] R. Rajkumar, M. Lübbering, T. Bell, M. Gebauer, B. Ulusay, D. Uedelhoven, T. Dilmaghani, R. Loitz, M. Pielka, C. Bauckhage and R. Sifa, “Automatic Indexing of Financial Documents via Information Extraction”, *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2021.
- [7] M. Lübbering, M. Gebauer, R. Ramamurthy, C. Bauckhage and R. Sifa, *Bounding open space risk with decoupling autoencoders in open set recognition*, *International Journal of Data Science and Analytics* (2022) 1.
- [8] M. Lübbering, M. Pielka, I. Henk and R. Sifa, “Datastack: unification of heterogeneous machine learning dataset interfaces”, *2022 IEEE 38th International Conference on Data Engineering Workshops (ICDEW)*, IEEE, 2022 66.
- [9] M. Lübbering, M. Gebauer, R. Ramamurthy, C. Bauckhage and R. Sifa, “From Open Set Recognition Towards Robust Multi-class Classification”, *Artificial Neural Networks and Machine Learning – ICANN 2022*, 2022.

Nomenclature

Datasets

- \mathcal{X} Dataset comprising samples $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}\}$
- \hat{p}_{data} Empirical data distribution, i.e., $\hat{p}_{\text{data}} \sim p_{\text{data}}$
- \mathbb{X} Sample space
- \mathbb{Y} Target space
- \mathbf{x} Sample out of \hat{p}_{data}
- C Set of classes present in p_{data} , i.e., $C = \{c_1, c_2, \dots, c_m\}$
- p_{data} Data generating process
- Y Targets $\{y^{(1)}, y^{(2)}, \dots, y^{(n)}\}$ corresponding to the samples within dataset \mathcal{X}
- y Target corresponding to sample \mathbf{x}

Neural networks

- α_j^k Activation, i.e., input of activation function in neuron j of layer k
- \hat{y} Prediction of a sample, i.e., $\hat{y} = f(\mathbf{x}; \Theta)$
- $\hat{\mathbf{x}}$ Reconstruction of sample \mathbf{x} , e.g., by an autoencoder
- Θ Neural network weights
- $\Theta^{(k)}$ Neural network weights in layer k
- $\varphi^{(k)}$ Layer k
- $\varphi_j^{(k)}$ Neuron j in layer k
- $\vartheta_{i,j}^{(k)}$ Neural network weight between neuron i in layer $k - 1$ and neuron j in layer k . $i = 0$ refers to the bias.
- $a(x)$ Activation function $a : \mathbb{R} \rightarrow \mathbb{R}$
- $d(\mathbf{x})$ Decoder part of an autoencoder
- $e(\mathbf{x})$ Encoder part of an autoencoder

$f(\mathbf{x}; \Theta)$ Neural network parameterized with weight matrix Θ

o Logit output o comprises the activations of the output layer, i.e., raw network outputs without postprocessing such as softmax.

Optimization

\mathcal{H} Hypothesis space

\mathcal{R} True risk

\mathcal{R}_{emp} Empirical risk as an approximation to \mathcal{R}

e_{MSE} Mean squared error

$f^*(\mathbf{x})$ Target function that we want to fit

L_{MSE} Mean squared error loss

Introduction

Machine learning methods, especially their subfield of deep learning, have become more and more prominent in our daily lives in recent years. Deep learning has a long-lasting history, with the initial perceptron formalization in the 50s [2] and the first principled training algorithm, namely backpropagation [3], dating back to the 80s. While the theoretical foundations of deep learning have been established early on, their further development and application to real-world problems were impeded for two reasons: lack of data availability and computing power. More specifically, training a deep neural network (DNN) for a simple classification task such as letter recognition requires thousands of examples and a significant amount of parallel computing power. Both requirements were still unmet in the early 90s, leading to unfulfilled expectations and major funding cuts (the artificial intelligence (AI) winter).

Today, the situation is entirely different. Powerful GPUs have entered the market, and the abundance of data due to the progression in digitalization has given rise to a resurgence in deep learning research and its various applications. Deep learning is now spearheading the advancements in machine learning research in many domains such as natural language processing (NLP) and computer vision. The architectures are becoming increasingly complex, demanding tremendous amounts of computational resources and training data. While the image classification architecture LeNet [4] had less than 3,246 trainable parameters 25 years ago, one of today's most complex language models, Palm, has a striking number of 540 billion parameters [5]. This transition from shallow neural networks to deep architectures has advanced state-of-the-art results in almost any machine learning domain, such as object detection [6], text-to-image synthesis [7], language modelling [8–10], named entity recognition [10, 11] and many others.

These impressive results have paved the way for ground-breaking achievements in which neural networks greatly surpassed the performance of human experts on challenging, non-repetitive tasks. In 2016, Google Deepmind released its sophisticated DNN system AlphaGo [12] that was trained to play Go, a game which favors intuition over pattern recognition and strategic way of playing due to a highly branching search space. In a series of three games, AlphaGo defeated the Go world champion of the time, in 2017. In 2020, Google Deepmind presented AlphaFold [13], a deep learning algorithm, which provided a solution to the 50 year-old, unsolved protein folding problem, showcasing the applicability of the field to computational biology and drug discovery [14]. Especially in the medical domain, where human failures are generally less tolerable, physicians have been outperformed on, e.g., breast cancer detection [15] by deep learning systems, transforming the mode of operation in medicine in the

foreseeable future [16].

Besides these breakthroughs, deep learning emerged into our daily lives in various, often hidden ways. For instance, the voice recognition in our phones or smart home devices uses deep learning to decipher our spoken words. Explicit and toxic content on social media is filtered by neural network-based computer vision and NLP models. Most larger-scale webshops track the click-level behaviour of every user accessing their website and use this information to recommend user-specific products to maximize the conversion rate. Financial institutions use neural networks to detect fraudulent behaviour or to predict their assets' price movements. In 2019, Google announced that it applies the bidirectional encoder representations from transformers (BERT [10]), a deep learning architecture, to understand the meaning within the English search queries submitted to their search engine¹. There is a general trend that, whenever there is an abundance of data and a repetitive task that can be solved using the information within said data, then this task has already, or soon will be solved by a machine learning algorithm [17].

While deep learning research has provided the algorithms and tools to establish machine learning in our everyday lives with outstanding success, there are also rising concerns regarding AI safety [18–20]. These concerns are backed by countless entries in the AI Incident database with multiple cases of human casualties [21], illustrating the necessity to scrutinize algorithm robustness, deployment scenarios and monitoring approaches. For example, when Microsoft released its Twitter Chat Bot Tay in 2016, Tay was maliciously tricked by Twitter users into making racist comments via its online learning approach. Another example of an AI failure was the adoption of a ball tracking algorithm in a game between the Scottish soccer teams Inverness Caledonian Thistle F.C and Ayr United FC in 2020, which mistakenly took a linesman's bald head as the ball². More severely, (semi-) autonomous driving has caused fatal crashes in the past due to failing machine learning models. For example, a pedestrian was fatally hit by one of Uber's self-driving test vehicles in 2018, as the system failed to detect jaywalking, according to the US National Transportation Safety Board (NTSB)³. In another case in 2016, a Tesla autopilot misidentified a truck's white trailer as the sky, leading to the car crashing into the truck, lethally injuring the driver, as reported by Tesla⁴. These examples suggest that only the deployment scenario's specific fault tolerance should ultimately determine the required level of algorithm robustness, human supervision and real-time monitoring.

All four of these examples also pinpoint a severe issue with machine learning and especially deep learning. Despite all models generalizing well within the concepts they were trained on, they did not anticipate the conceptual shift and the fat tail of unseen conditions when deployed in the real world. Technically, the optimization problem formulation of DNNs only involves the separation of observed classes and disregards possible exposure to outliers and dataset shift at inference time [22]. Thus, no principled mechanism in the training procedure controls the model's behaviour in these scenarios, leading to severe robustness deficiencies, and opening the door for disastrous outcomes. Throughout this thesis, we define dataset shift as a significant difference in the joint distributions (i.e., input and target) at training and test time. We leverage samples of classes unseen during training as a proxy to estimate dataset shift robustness.

In this work, we target the robustness problem for DNNs and aim to increase their robustness

¹<https://blog.google/products/search/search-language-understanding-bert/>

²<https://www.thescottishsun.co.uk/sport/football/6216560/inverness-fans-camera-linesman-bald-head-ball-ayr/>

³<https://www.nts.gov/investigations/accidentreports/reports/har1903.pdf>

⁴<https://www.tesla.com/blog/tragic-loss>

to unseen events by teaching the network awareness of the unknown by reformulating the training objective. Whenever the model is exposed to out-of-distribution (OOD) samples, i.e., samples that cannot be explained by the inlier distribution of the training data, we want the model to be unsure about their class affiliation.

To achieve this challenging goal, we turn towards the semi-supervised, autoencoder-based outlier detection method, *one-class autoencoder* (OCA), since this method is designed to filter out-of-distribution data. Essentially, OCA minimizes the reconstruction error of the inlier data and uses the reconstruction error as a signal for the outlierness of a sample at inference time.

Following the intuition behind the manifold hypothesis, natural data generally occupies a low dimensional manifold within a higher dimensional space. For instance, the image of a face can be represented by distinguishing features such as shape of eyes, nose, mouth, etc. rather than a high dimensional 3-channel image. An autoencoder, with its architectural bottleneck and reconstruction objective, aims to learn this latent manifold within its embedding space, which is the natural representation of the data generating process under the manifold hypothesis. Any deviation from the latent manifold can be treated as an outlier.

Interestingly, we find that the semi-supervised training approach leads to poor classification performance, especially when the outliers correlate with the inliers in feature space. Therefore, we propose a more aggressive approach, namely an *adversarially trained autoencoder* (ATA) [23, 24], which additionally considers the outliers at training time by maximizing their reconstruction errors to make the reconstruction error an even better signal of outlierness. This supervised outlier detection approach is the foundational basis of this work. We find that leveraging outliers at training time allows the network to learn the distinctive, application-specific properties of outliers, demonstrating ATA’s superiority over semi-supervised outlier detection methods. Finally, ATA resembles our first milestone towards robust classification in deep learning by providing a principled formulation of the unknown.

We follow up on ATA by proposing an adaptation of ATA, resulting in the open set recognition (OSR) [25] method *decoupled autoencoder* (DAE) [22]. In contrast to outlier detection, the objective of OSR is to distinguish the observed inliers from partially observed rest classes which can comprise samples from the inlier problem domain and OOD data. The adversarial training approach introduced with ATA excels in this setting as it jointly optimizes for classification performance and outlier detection. Our results demonstrate that DAE outperforms current state-of-the-art OSR baselines.

Finally, building upon DAE, we composed multiple DAEs as ensemble components within our proposed Informer architecture for robust multi-class classification [26]. Each of the k ensemble components learns a one-vs-rest (OVR) relationship for one of the k inlier classes and is thereby capable of rejecting unknown samples. By design, the Informer differentiates between aleatoric and epistemic uncertainty, a crucial robustness and interpretability criterion [27].

In the remainder of this chapter, we introduce the general training process of deep neural networks. We then derive why these networks are only well-equipped for closed-set classification tasks and do not generalize to unseen outliers. Afterwards, we introduce supervised outlier detection and disambiguate the difference from classification and unsupervised outlier detection. We then bridge the gap to open set recognition by shedding light on the connection to outlier detection, followed by an introduction to manifold learning with autoencoders. Finally, we provide a detailed, chapter-wise outline of this work.

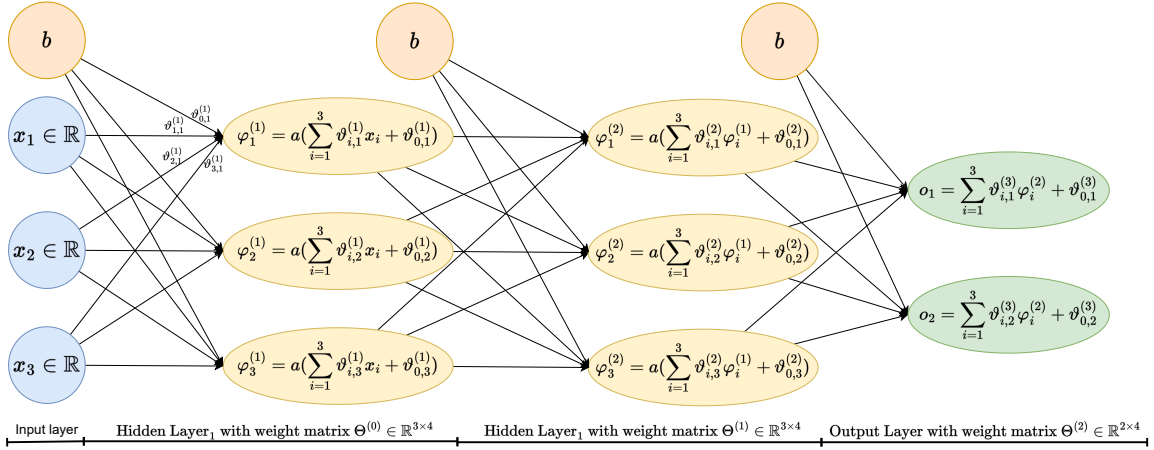


Figure 1.1: Exemplary architecture of a feedforward neural network: Each neuron $\varphi_i^{(k)}$ in layer k calculates the weighted sum of the output in the preceding layer and the weighted $b = 1$ and subsequently maps it onto a scalar value via a non-linear activation function $a : \mathbb{R} \rightarrow \mathbb{R}$. The final output layer o predicts a sample as one of the classes according to the maximum output value. These outputs are also referred to as logits and are usually mapped to probability scores via the sigmoid function in the case of binary classification or the softmax function in case of multi-class classification. The objective during DNN training is to adapt the weights such that the true target function $f^* : \mathbb{X} \rightarrow \mathbb{Y}$ is approximated accurately.

1.1 Optimization of Deep Neural Networks

A feedforward neural network, also referred to as a multi-layer perceptron (MLP), is an acyclic, directed graph with a non-linear function within each node mapping the weighted sum of the input to a scalar output value. The resulting nested chain of non-linear functions is a universal function approximator of the true target function $y = f^*(\mathbf{x})$ [28]. Since the target function / data generating process p_{data} is generally unknown, a dataset $\mathcal{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}\}$ with sample $\mathbf{x}^{(i)} \in \mathbb{R}^n$ is sampled from the empirical data distribution \hat{p}_{data} instead to approximate $f^*(\mathbf{x})$. Fig. 1.1 illustrates an MLP with two hidden layers each comprising three neurons. As we pass an input sample \mathbf{x} to the first layer ($k = 1$), the activation $\alpha_j^{(1)}$ of neuron $\varphi_j^{(1)}$, i.e., the j^{th} neuron within layer 1, computes to

$$\alpha_j^{(1)}(\mathbf{x}) = \sum_{i=1}^n \vartheta_{i,j}^{(1)} x_i + \vartheta_{0,j}^{(1)}, \quad (1.1)$$

where $\vartheta_{i,j}^{(1)}$ refers to the weight on the connection from input \mathbf{x}_i to neuron j in layer 1. The weight $\vartheta_{0,j}^{(k)}$ is referred to as the bias which translates the riff of the non-linear activation function a . The activation function is a non-linear mapping $a : \mathbb{R} \rightarrow \mathbb{R}$ which prevents the neural network from collapsing into a linear model. Commonly used activation functions are the sigmoid function $a_s(x) = \frac{e^x}{e^x + 1}$ or the rectified linear unit (ReLU) $a_r(x) = \max(0, x)$ [29]. The forward pass within the hidden layers $\varphi^{(k)}$ with $k > 1$ can be formulated recursively for a given neuron $\varphi_j^{(k)}$ by

$$\varphi_j^{(k)} = a \left(\sum_{i=1}^{|\varphi^{(k-1)}|} \vartheta_{i,j}^{(k)} \varphi_i^{(k-1)} + \vartheta_{0,j}^{(k)} \right), \quad (1.2)$$

$|\varphi^{(k)}|$ denotes the number of neurons in layer k . The raw output vector o of the network, also referred to as logits, is calculated as the weighted sum of the activations of the last hidden layer, as shown in Fig. 1.1. If there is only a binary output, i.e., $o \in \mathbb{R}$, then the logit can be scaled to $[0, 1]$ via the sigmoid function to obtain class probabilities. In case of multi-class classification with classes $C = \{c_1, c_2, \dots\}$, the output vector $\mathbf{o} \in \mathbb{R}^{|C|}$ is usually transformed to a probability distribution via the softmax function

$$\text{softmax}(\mathbf{o})_i = \frac{e^{o_i}}{\sum_{j=1}^{|C|} e^{o_j}}, \quad (1.3)$$

with $\sum_{i=1}^{|C|} \text{softmax}(\mathbf{o})_i = 1$.

Given this definition of a feedforward neural network for binary and multi-class classification, the network is a chain of nested, differentiable functions. When training such a network, our objective is to accurately approximate the target function $f^* : \mathbb{X} \rightarrow \mathbb{Y}$, mapping the sample space \mathbb{X} to the target space \mathbb{Y} . Partially following the generally accepted notation and derivation of DNN optimization in [17], the target function can also be regarded as a data generating process $p_{\text{data}}(\mathbf{x}, y)$ that we aim to approximate. An optimal approximation would minimize the risk \mathcal{R}

$$\mathcal{R}(\Theta) = \mathbb{E}_{(\mathbf{x}, y) \sim p_{\text{data}}} L(f(\mathbf{x}; \Theta), y), \quad (1.4)$$

where (\mathbf{x}, y) denotes a sample/target pair sampled i.i.d. from p_{data} and L denotes the per-sample loss function.

In machine learning, the target function must be approximated by the empirical data distribution $\hat{p}_{\text{data}}(\mathbf{x}, y)$, i.e., a set of observed samples, since the data generating process p_{data} is generally unknown [17]. Thus, the overall optimization problem must instead be formulated as

$$\mathcal{R}_{\text{emp}}(\Theta) = \mathbb{E}_{(\mathbf{x}, y) \sim \hat{p}_{\text{data}}} L_{01}(f(\mathbf{x}; \Theta), y) \quad (1.5)$$

$$\Theta^* = \arg \min_{\Theta} \mathcal{R}_{\text{emp}}(\Theta), \quad (1.6)$$

which corresponds to the minimization of the empirical risk \mathcal{R}_{emp} concerning the 0-1 loss L_{01} w.r.t. the network weights Θ of neural network f . The 0-1 loss is defined as $L_{01}(\hat{y}, y) = \mathbb{1}(y \neq \hat{y})$ and by specification, is non-differentiable, making L_{01} inapplicable for gradient-based optimization of DNNs.

In practice, we optimize the model weights in an iterative fashion by calculating the loss gradient w.r.t. the weights, a process referred to as (stochastic) gradient descent:

$$\Theta \leftarrow \Theta - \gamma \frac{\partial L(f(\mathbf{x}; \Theta), y)}{\partial \Theta} \quad (1.7)$$

The partial derivatives guide the optimization of the weights Θ at a learning rate γ into minimizing the overall loss. In the past years, improved optimizer methods have been proposed that adapt the learning rate for each model weight internally. These optimizers show better generalization performance, as they are less prone to local minima/saddle point convergence and support sparse data [30]. Throughout this thesis, we prefer these superior optimizers (e.g., Adadelta [31] and Adam [32]) over stochastic gradient descent (SGD).

As a solution to the non-differentiable L_{01} loss, a differentiable surrogate loss such as cross-entropy is employed. The minimization of this loss not only minimizes \mathcal{R}_{emp} implicitly but also enforces better separation between the classes resulting in better generalization performance by further reducing risk

\mathcal{R} [17].

The foundations of cross-entropy loss base in the maximum likelihood framework. Partially following the derivation steps and generally accepted notation of [17], the maximum likelihood estimator is given by

$$\Theta^* = \arg \max_{\Theta} p_{\text{model}}(\chi | \Theta), \quad (1.8)$$

whose objective is to select the hypothesis $\Theta \in \mathcal{H}$ from the hypothesis space that maximizes the likelihood of the data χ . This means that the hypothesis Θ^* explains the data best. We can reformulate the maximum likelihood estimator in a supervised setting as the product of the likelihoods of i.i.d. samples given model parameters Θ

$$\Theta^* = \arg \max_{\Theta} \prod_{i=1}^{|\chi|} p_{\text{model}}(y | \mathbf{x}^{(i)}; \Theta), \quad (1.9)$$

where $|\chi|$ denotes the dataset size and $\mathbf{x}^{(i)}$ the i^{th} sample within χ . Due to numerical instability of the product, the maximum likelihood is reformulated as a minimization of the negative log likelihood

$$\Theta^* = \arg \min_{\Theta} -\mathbb{E}_{(y, \mathbf{x}) \sim \hat{p}_{\text{data}}} \log p_{\text{model}}(y | \mathbf{x}; \Theta). \quad (1.10)$$

The negative log likelihood is equivalent to the cross-entropy H between the data and the model predictions:

$$H(\hat{p}_{\text{data}}, p_{\text{model}}) = - \sum_{i=1}^{|\chi|} \hat{p}_{\text{data}}(y | \mathbf{x}^{(i)}) \log p_{\text{model}}(y | \mathbf{x}^{(i)}; \Theta) \quad (1.11)$$

$$= - \frac{1}{|\chi|} \sum_{i=1}^{|\chi|} \log p_{\text{model}}(y | \mathbf{x}^{(i)}; \Theta) \quad (1.12)$$

$$= -\mathbb{E}_{(\mathbf{x}, y) \sim \hat{p}_{\text{data}}} \log p_{\text{model}}(y | \mathbf{x}; \Theta) \quad (1.13)$$

Thus, maximizing the likelihood within the data w.r.t. the model parameters Θ corresponds to minimizing the cross-entropy. Naturally, as $\lim_{|\chi| \rightarrow \infty} p(\mathbf{x} | \Theta) = p_{\text{data}}(\mathbf{x})$, the empirical risk \mathcal{R}_{emp} and true risk \mathcal{R} become equivalent and the minimization of cross-entropy results in the minimization of \mathcal{R} . Technically, the minimization of cross-entropy is performed via gradient descent, as there exists no closed-form solution due to the non-convexity of the DNN optimization problem.

However, maximum likelihood estimation and empirical risk minimization only generalize under the very strong assumption that \hat{p}_{data} resembles the true data generating process p_{data} . In the case of an unobserved fat tail of p_{data} , e.g., outliers, noise, or dataset shift at inference time, the model's behavior is unpredictable as these samples were not reflected in the maximum likelihood estimation. It has been shown by [18, 33–37], that a DNN provides overly confident estimates on OOD samples, even though we expect a well-calibrated model to provide predictions of high uncertainty.

The issue of unforeseeable behavior under OOD exposure is illustrated in Fig. 1.2. An MLP is trained to differentiate the red and blue classes for each of the four toy datasets, resulting in the green decision boundaries. The optimization process maximizes the likelihood of the data and enables the model to generalize within the concepts of the training data. When the model is exposed to noise at

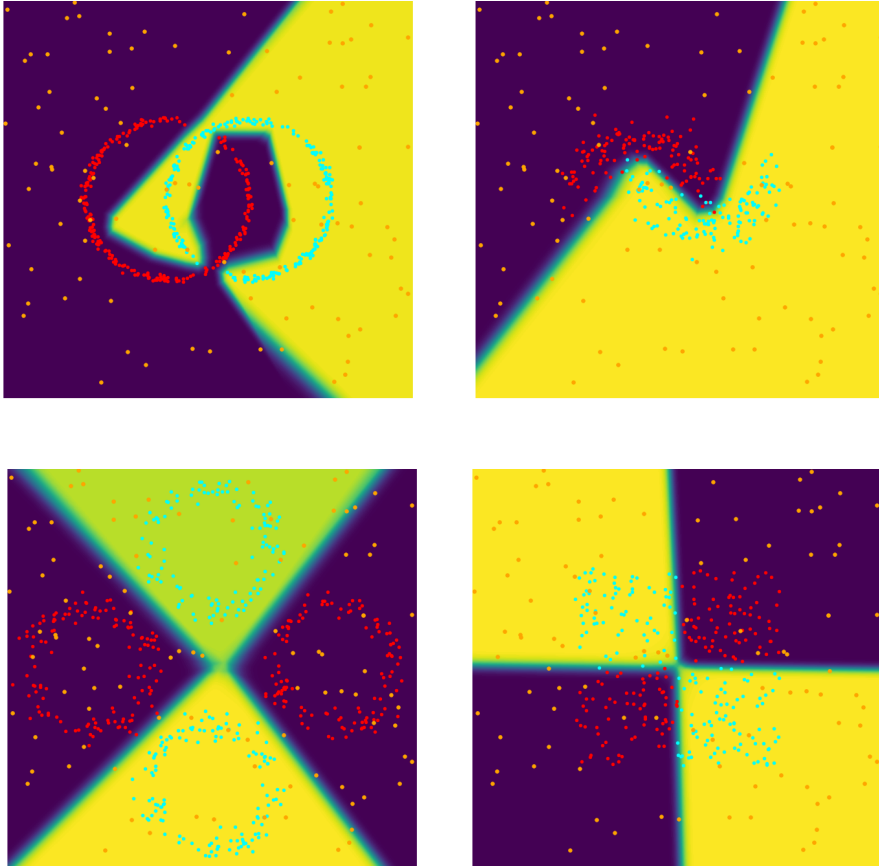


Figure 1.2: Demonstrating the intrinsic issues of empirical risk minimization in DNNs: We trained an MLP on each binary toy dataset (red and blue samples) and highlighted the decision boundary surface. While the model generalizes well within the classes it was exposed to at training time, each noise sample (orange) is attributed to one of the two classes with high confidence, jeopardizing model robustness. We would expect a robust model to learn a hull around the inlier data (i.e., red and blue samples), efficiently rejecting any outlier data at test time. We will propose such methods based on reconstructive representation learning in this thesis.

test time, as indicated by the orange samples, these samples are falsely predicted as one of the two classes with maximum confidence.

Even though the empirical risk minimization is a well-founded approach when the training data reflects the true data generating process, the reported issue of overly confident predictions on OOD data and the decision boundary surfaces in Fig. 1.2 pinpoint a fundamental issue concerning model robustness. In our work, we turn towards outlier detection methods whose objective is to identify samples that cannot be attributed to \hat{p}_{data} , i.e., outliers. One of the central ideas in this work is the combination of outlier detection methods with classification methods to jointly optimize robustness and empirical risk. In safety-critical environments, such robust methods are of utmost importance for preventing AI accidents.

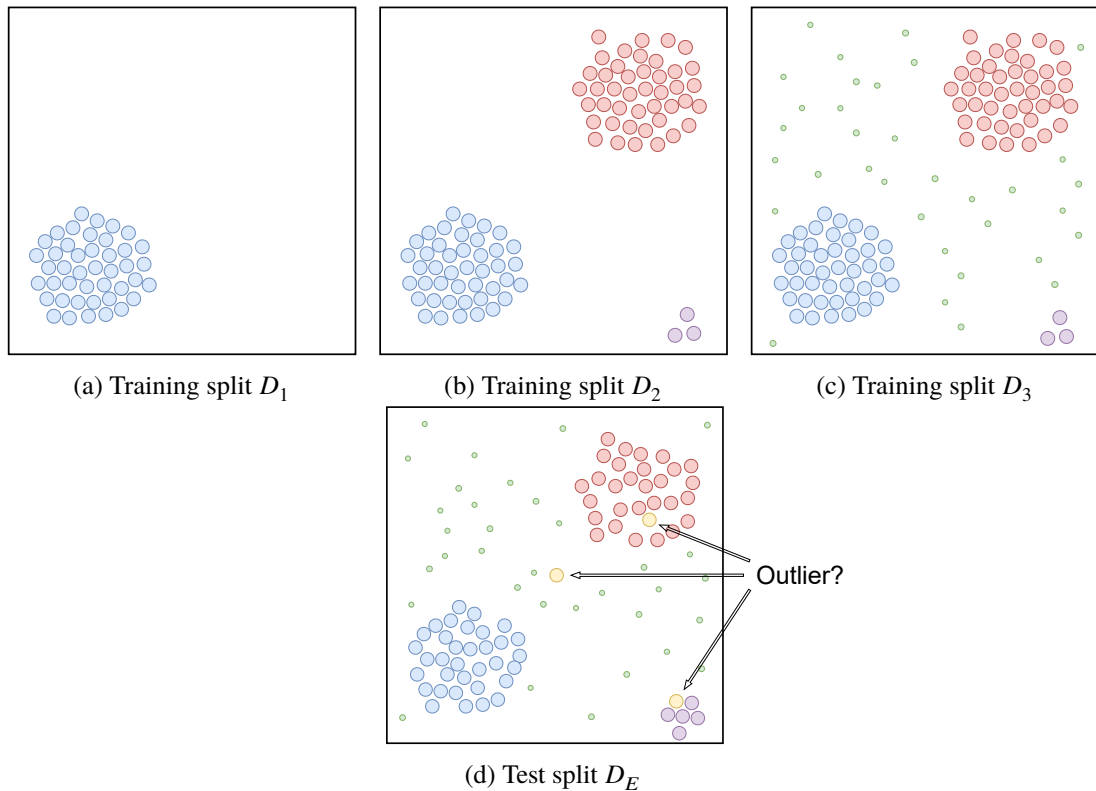


Figure 1.3: Different perspectives on the outlieriness of a sample: Training set D_1 contains only a single inlier class, training set D_2 contains three disjoint classes and training split D_3 contains noise additional to D_2 samples. Given the three different training sets D_i , which of the samples highlighted in orange within test set D_E qualify as outliers? The underspecification of the problem allows for different conclusions. Example inspired by [43].

1.2 Introduction to Outlier Detection

Outlier detection has a long-lasting history of research dating back to the 60s [38]. It is still a largely unsolved research domain with various open subproblems touching various related fields in machine learning, such as open set recognition [39], one-class classification [40] and novelty detection [41].

In his book on outlier detection [42], Hawkins defines an outlier as "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism". This abstract outlier definition is inherently ambiguous, as outliers have a strong contextual dependency.

The contextual dependency can be understood from the inliers in Fig. 1.3, providing a data-centric description of the normality. Given split D_1 , it is reasonable for an outlier detection model to learn a normality representation from the inlier distribution highlighted in blue. Any significant deviations from this distribution can be understood as outliers, resulting in the three isolated samples, highlighted in orange, being predicted as outliers in D_E . For D_2 , the orange sample within the red distribution is no longer an outlier, as its origin can be directly associated with the red class. Given the noise in D_3 , the outlier attribution of the remaining two samples becomes ambiguous. Depending on the context, the green noise and the infrequent purple class might not raise enough suspicions to qualify as outliers.

The example⁵ made by Prof. Bauckhage in his pattern recognition lecture at the University of Bonn illustrates the general underdetermination of outlieriness. Given a set of composers {Stockhausen, Bach, Grieg, Beethoven, Brahms, Wagner}, the question *Which composer qualifies as an outlier?* is completely underdetermined. For instance, if clustered by genre, Stockhausen would stand out as the only non-classical composer, whereas if clustered by nationality, Grieg, as a Norwegian, would be the only non-German composer. This example highlights the different objectives of supervised and unsupervised outlier detection. Unsupervised outlier detection aims to detect generic saliencies, independent of the concrete problem domain. These saliencies are often forwarded to experts for final review [43]. In contrast, supervised outlier detection leverages the annotated outliers as guidance to learn, not only the concept of normality, but also the concept of problem specific outliers [43], often rendering expert review redundant. The related imbalanced classification task is often considered a subproblem of supervised outlier detection.

The problem of outlieriness underdetermination is further amplified by the data sparsity induced by high-dimensional data, in combination with noisy/irrelevant features [44–46]. These settings make outlier detection a challenging task, as the outlier scores of such detectors become uninformative and sample pairs become equidistant, irrespective of their true classes [47, 48]. Therefore, it is always advised to utilize labeled outliers during model training, as they uncover the non-linear outlier properties hidden within the high-dimensional data [43].

The aforementioned complexities associated with outlier detection are present in many practical classification tasks, such as fraud detection [49], intrusion detection [50], surveillance, monitoring, predictive maintenance [46], and medicine [51, 52]. This is why outlier detection has significantly contributed to these fields [43, 45, 46].

1.2.1 Evaluating Outlier Detection and Open-set Recognition Systems

The evaluation procedure of outlier detection systems largely depends on the availability of annotated data. In the case of unsupervised (trained and evaluated on unlabeled data, containing normal and anomalous instances) and semi-supervised outlier detection (trained and evaluated solely based on inlier classes), "goodness of fit" metrics based on cluster/sample similarity can be applied to measure the coherence of the clusters and inlier samples [43].

With the availability of ground truth data on normal and anomalous classes, traditional classification performance metrics can be calculated in the supervised setting, though, special attention needs to be paid to inherent class imbalances. In particular, the low base rate of outliers allows for reaching close to perfect scores on accuracy metrics [44]. Thus, outlier detectors are often manually evaluated based on their receiver operating characteristic (ROC) curve or precision-recall (PR) curve. The models can be compared irrespective of any discrete decision boundary by calculating the area under such curves, i.e., AUROC and AUPR, respectively. This is an important evaluation criterion as the decision boundary is usually determined based on the deployment domain, accounting for the cost of false positives and false negatives.

More specifically, the AUROC metric calculates the area under the ROC curve, which maps the false positive rate (FPR) onto the true positive rate (TPR) for each threshold. The two rates are defined by $FPR = \frac{FP}{FP+TN}$ and $TPR = \frac{TP}{TP+FN}$, where TP, FP, TN and FN depict the number of true positives, false positives, true negatives and false negatives at a certain threshold, respectively. Note that TPR

⁵<https://youtu.be/-vsCj0ijP4U?t=547>

is also often referred to as recall in the literature. From an interpretation point of view, AUROC yields the probability of a random test sample being ranked higher than a random inlier [33, 53], and therefore is invariant to class imbalance. This invariance to class imbalance allows for interpretable results across datasets, since a random classifier achieves an expected AUROC score of 50%, and a perfect classifier, a score of 100%.

Since AUROC in isolation can be insufficient for outlier detection and open-set recognition with their prevalent class imbalance, we can additionally consider AUPR. This metric is generally employed when faced with the "needle in the haystack problem", as AUPR takes the different class base rates into account [33]. Similar to AUROC, the PR curve maps the recall onto the precision $= \frac{TP}{TP+FP}$ for each threshold. Since the AUPR metric is base rate dependent, there is no fixed baseline AUPR score for a random classifier like there is for AUROC. In fact, given a random classifier, the AUPR score is roughly equal to the random classifier's precision, which is equal to the rate of the positive class [33, 54]. As a result, it is essential to always communicate the AUPR scores w.r.t. the random classifier performance and the pre-defined positive class; otherwise, the metric is difficult to interpret meaningfully.

1.2.2 Traditional Outlier Detection Methods

From a methodological point of view, traditional outlier detectors are based on probabilistic models, linear regression models / PCA and proximity-based models [43], and are implemented in an unsupervised fashion. Traditional outlier detection mostly disregards supervised methods due to the frequent lack of outlier labels in practice.

The foundational objective of each method is to learn a representation of normality, allowing the modeling of abnormality as a deviation from said representation. The main distinction between these outlier detection methods lies within the implementation of learning the outlierness score.

Probabilistic models such as Gaussian mixture models maximize the likelihood of the training data by variation of the model parameters, e.g., by applying the expectation maximization (EM) algorithm. Samples are predicted as outliers when they cannot be explained by any of the learned Gaussian distributions [43, 55, 56]. Such models have strong prior assumptions, as the number of mixture components and type of distribution need to be determined without access to actual outliers.

Linear models such as principal component analysis (PCA) and linear regression aim to fit the data to a set of linear parameters by minimizing, e.g., the mean square error (MSE)[43]. These models are interpretable and provide simple means for noise reduction, however, due to their linearity, the detection of more complex, non-linear outliers is limited. This has led to the development of kernel PCA and kernel SVM, which by design are capable of detecting suspicious, non-linear patterns [43].

Proximity-based models such as k-nearest neighbors (KNN) or k-Means clustering measure the distance to reference points used as heuristics for the outlierness of a sample [43, 57]. While these heuristics work reasonably well on simple outlier detection tasks, the outlierness scores get disturbed significantly with the advent of noise [43]. Building on the idea of distance-based outlier scores, the tail of the outlierness score distribution can be fit to extreme value distributions [43], such as the Weibull distribution [58].

While all of these traditional outlier detection methods have proven successful within various practical applications, they also share various shortcomings. Most significantly, none of these methods incorporates annotated outliers in the training procedure, hence, often failing to convey the outlier properties of interest. This is often further impaired by noisy, high-dimensional settings, preventing

the model from detecting the non-linear outlier relationship [46]. Therefore, in the following section, we discuss supervised outlier detection with its different challenges and advancements in the context of deep learning.

1.2.3 Deep (Supervised) Outlier Detection

Supervised outlier detection, with its intrinsic class imbalance, induced by the low base rate of outliers, can be regarded as a challenging subtype of supervised classification. Further, complicating matters, only a subset of the outlier classes may be available at training time, transferring the aforementioned challenges of semi-supervised outlier detection to supervised outlier detection.

In practice, the class imbalance often leads to classifiers, trivial or sophisticated, reaching almost perfect accuracy scores by always predicting the majority class. This issue can be counteracted by cost-sensitive training objectives and over-/undersampling of the outlier/inlier classes [43, 59]. Technically, due to the iterative training in deep learning, cost-sensitive learning and over-/undersampling can be implemented in the loss function and data loader, respectively. Another approach is sampling linear combinations of outlier pairs to simulate more diverse outlier samples, as proposed by the SMOTE algorithm [60]. Despite its long history, imbalanced classification is still subject to research, and remains an unsolved problem to this day [61].

To learn the non-linear characteristics of outliers, scholars have conducted research on a plethora of, mainly unsupervised, deep learning architectures [45, 46]. At their core, these architectures adopt the idea from traditional outlier detectors of learning a representation of normality to classify substantial deviations from the normal representation as outliers. Outlier detection based on deep learning can be categorized into two research directions [45]. Firstly, transfer learning can be utilized to reduce the high dimensional input space by, e.g., using the latent activations of a pre-trained model [62] or the latent state of an autoencoder [62, 63]. In this case, outlier detection methods are applied to the lower-dimensional representations as a downstream task, decoupling the representation learning from the outlier detection. Thus, these approaches can only work under the assumption that the representations learned during pre-training capture the outlierness of a sample, despite both being possibly disjoint objectives.

In the second direction, end-to-end outlier detectors learn a representation of normality using representation learning. In the case of autoencoders [64, 65], which minimize the reconstruction error on the inlier data (semi-supervised), or the entire dataset with underrepresented outliers (unsupervised), the reconstruction error becomes predictive of the inlierness of a sample due to the bottleneck limiting information flow. This approach assumes that the input space is mostly free of noise and ambiguous features, which would dilute the reconstruction error-based outlierness signal otherwise. Generative adversarial networks (GANs), with their generator and discriminator being trained in a min-max fashion, can also be utilized for outlier detection [66]. In AnoGAN, a GAN-based outlier detector, the outlierness of a sample is defined as a linear combination of the residual loss (difference between input sample and generated sample) and the discrimination loss (i.e., classification loss within the discriminator).

Surprisingly, although the incorporation of outliers is strongly suggested if available [43], the two research directions within deep outlier detection generally do not incorporate outliers into the training process and mainly focus on unsupervised [62, 63, 66] or semi-supervised [67, 68] methods. Our results on supervised outlier detection in Ch. 3 and Ch. 4 clearly show that incorporating outlierness information during training, enhances the model's capabilities for detecting application-specific

outliers significantly.

1.2.4 Relationship Between Outlier Detection, Imbalanced Classification, Open Set Recognition, One-class Classification, Novelty Detection, and One-vs-rest Classification

Outlier detection, i.e., the task of detecting samples which cannot be attributed to an inlier data generating process, plays a crucial role within robust classification, as a robust classifier should be able to reject such anomalies. This section briefly introduces the related concepts and elaborates on the connection to outlier detection.

Due to the generally low base rate of outliers, supervised outlier detection faces severe class-imbalance. Therefore, imbalanced classification can be regarded as a subproblem of supervised outlier detection. Novelty detection is often treated as a semi-supervised setting, in which the model is not exposed to the novel outlier types (novelties) at training time.

One-vs-rest (OVR) classification [69], as a supervised classification setting, aims to distinguish a single class of interest (COI) from a set of observed rest classes (RCs). This closed-set classification scenario neglects the existence of outliers and is often subject to severe class imbalance.

One-class classification (OCC) can be regarded as a semi-supervised setting which exposes the model only to COI samples at training time. At inference time, the model is supposed to reject samples deviating from the COI representation [68, 70], i.e., outliers and noise, but also any classes that are similar to the COI. The lack of outlier exposure limits the model to unsupervised or semi-supervised outlier detection methods.

Finally, open set recognition (OSR) [25] exposes the model to a set of COIs (i.e., closed set) and a set of observed rest classes (i.e., the observed subset of the open set) at training time. The objective is to leverage this information to learn a decision boundary separating the closed set from the open set. Note that the open set comprises observed RCs and unobserved samples belonging to unseen RCs, noise, or outliers. Open set recognition formulates an objective that jointly minimizes the open space risk and binary classification error between COIs and RCs. The open space risk can be interpreted as the slack between the learned decision boundary and the tightest hull wrapping the COIs. Minimizing the open space risk implicitly also minimizes the risk of false positives, resulting in superior model robustness.

1.3 Autoencoders From a Manifold Learning Perspective

The bottleneck in the center of an autoencoder is often considered to be a lossy compression, limiting information flow. Accordingly, the network is forced to find a non-linear function mapping the input data to a compressed representation in the hidden space, from which the input can be reconstructed.

Another, more in-depth perspective can be derived from the manifold hypothesis. An n -dimensional manifold is defined as a topological space that is locally Euclidean, i.e., homeomorphic to the Euclidean space \mathbb{R}^n [71]. For instance, a sphere in \mathbb{R}^3 is a two-dimensional manifold, since the two-dimensional neighborhood of a given point on the manifold is Euclidean. The manifold hypothesis argues that the high dimensional data we obtain from our natural environment can be represented on a manifold within a lower dimensional space [69]. For instance, a 28×28 MNIST image $\mathbf{x} \in \mathbb{R}^{784}$ contains vast amounts of redundant pixel information, which could be compressed into features such as strike

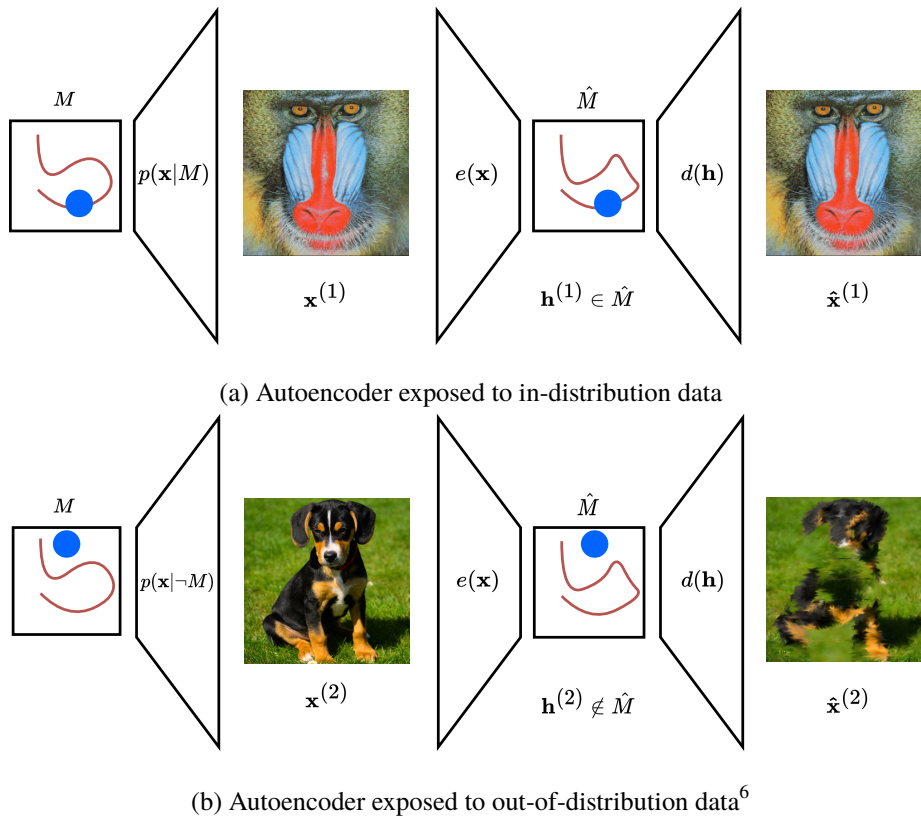


Figure 1.4: Illustration of autoencoder-based manifold learning by the example of monkey face detection: In Fig. 1.4(a), the image is obtained from the natural manifold M and projected into the high dimensional image space. The autoencoder trained to reconstruct monkey face images encodes the image $\mathbf{x}^{(1)}$ by mapping it onto the learned manifold \hat{M} , to subsequently reconstruct $\mathbf{x}^{(1)}$ as $\hat{\mathbf{x}}^{(1)}$ from the low dimensional embedding $\mathbf{h}^{(1)}$. Note that the sampled image lies in \mathbb{R}^n with $n \gg 2$ and the manifold is one-dimensional embedded within \mathbb{R}^2 . In Fig. 1.4(b), the input image does not originate from M and is therefore not placed onto \hat{M} by the encoder e and cannot be reconstructed.

width and strike length on a low dimensional manifold. It follows that an i.i.d. sample $\mathbf{x} \sim \mathbb{R}^{784}$ is highly unlikely of being mapped onto the MNIST manifold, as it would have to follow the distinctive MNIST patterns. Conversely, there is data that cannot be generated from a low dimensional manifold. Consequently, manifold learning poses an effective angle for outlier detectors.

Autoencoders rely heavily on the manifold hypothesis. They aim to learn two non-linear mapping functions. The encoder maps the high dimensional data onto a learned manifold within the lower dimensional embedding space that is similar to the natural manifold, and the decoder reconstructs the original, high dimensional input from the low-dimensional embedding in the manifold. In this framework, the central idea of autoencoder-based outlier detection is that the decoder maps the data generating process from the input space onto the natural manifold and the decoder accurately reconstructs the embeddings. When the model is exposed to outliers, i.e., samples not associated with the data generating process, the outlier is expected to not be placed onto the manifold, and consequently,

⁶The author would like to thank Eddi for being such a good boy.

not be reconstructed accurately. This concept is outlined in Fig. 1.4 by the example of reconstructing monkey portraits. Under the condition that monkey portraits occupy a low dimensional manifold M , we sampled the high dimensional portrait in Fig. 1.4(a) from M . During training, the encoder learned a similar manifold $\hat{M} \simeq M$ to represent monkey portraits, allowing the decoder to accurately reconstruct the images. When the image does not occupy the natural manifold, as shown in Fig. 1.4(b), the encoder maps the sample far from \hat{M} in embedding space, leading to poor reconstruction.

As we will show empirically, semi-supervised autoencoders tend to learn a manifold that in the limit mimics the identity function $f(\mathbf{x}) = \mathbf{x}$. As a solution, supervision can guide autoencoders to learn a manifold solely for the inlier data generating process and force any deviations to be placed far from the manifold in the embedding space.

1.4 Robust multi-class classification

Model robustness is a crucial concern that can be considered orthogonal to accuracy. An accurate model generalizes well on the concepts it was trained on, meaning the accuracies achieved on a training split extend to a hold-out test split that is representative of p_{data} . However, when p_{data} is subject to corruptions such as noise, a robust model is supposed to reject such anomalies as unseen conditions.

In this work, we consider three different robustness aspects. Firstly, when deployed in the open world, a model can be exposed to natural corruptions (e.g., outliers or noise) that were unobserved during training [35, 72, 73]. It has been shown by [33], that traditional DNNs falsely predict such corruptions as one of the inlier classes with high confidence, jeopardizing model robustness. Secondly, with adversarial examples (engineered samples to fool a DNN), there is another type of corruptions a model is supposed to reject [37, 74]. There is a plethora of adversarial attacks based on gradient ascent and black box testing, aiming to find samples with minimal deviations from true samples that flip the class prediction [75, 76]. A robust model is capable of rejecting these corruptions by associating the sample with none of the COIs, e.g., by predicting the sample as a uniform class distribution.

Finally, calibration accuracy can be placed under the wing of robustness [35, 77–79]. The confidence predicted by a model should resemble the model’s true error rate. For instance, a model predicting a sample with 90% confidence should be correct in 90% of the cases. It is important to note that calibration performance is orthogonal to accuracy [35]. A random model can be perfectly calibrated while being inaccurate.

In this work, we develop algorithms that are robust w.r.t. to these three robustness concerns.

1.5 Outline of This Work

On a high level, this work aims to map out the means to teach DNNs sensitivity to the unknown in a supervised (multi-class) classification setting. As mentioned previously, awareness of the unknown, and more technically, model robustness, are critical requirements for improved AI safety, with countless applications. To reach this goal, our work can be divided into the following three algorithmic milestones:

1. Extension of semi-supervised outlier detection towards supervised, deep learning-based outlier detection of application-specific anomalies

2. Utilization of supervised outlier detectors for robust one-vs-rest classification in an open-world setting, i.e., open set recognition (OSR)
3. Extension of our proposed OSR methods to robust multi-class classification

Algorithmically, our idea is founded on traditional, semi-supervised, autoencoder-based outlier detectors. Firstly, we extend this approach towards the supervised outlier detection method, *adversarially trained autoencoder* (ATA), by enriching the objective with concrete outlier information. Afterwards, we adapt ATA towards open set recognition by teaching it the decision boundary in reconstruction error space in an end-to-end fashion, as well as, enabling the resulting *decoupled autoencoder* (DAE) method to estimate the (subjective) prediction uncertainty. The DAE training objective forces the model to learn a compact decision boundary around the inlier samples in reconstruction error space, resulting in a decreased false positive rate. Finally, we arrange DAE in an ensemble, where each component learns a one-vs-rest relationship for a different class. This results in a robust multi-class classification algorithm that captures different types of uncertainties, improving the interpretability of model uncertainty, a key concern of AI safety.

To the best of our knowledge, no one has previously worked on this road map. While multiple papers have stated the general possibility of leveraging outliers to increase corruption robustness [34, 80], our contribution is based on successfully connecting outlier detection and robust multi-class classification in a principled manner. Specifically, we propose three novel algorithms which outperform state-of-the-art methods in (supervised) outlier detection, open set recognition and robust classification.

The chapter-wise outline of this work is structured in six parts. In Ch. 1, we introduced the reader to DNN optimization, outlier detection, robust classification and provided the theoretical foundation to comprehend our theoretical contributions. In Ch. 2, we present the two open-source frameworks *Datastack* and *MLgym*, which build the infrastructure for our experimental setup.

With these two frameworks, we implement an end-to-end machine learning research pipeline supporting efficient prototyping, reproducibility of results and insightful evaluation:

- **Datastack**⁷: The heterogeneous data format landscape of raw machine learning datasets complicates well-designed end-to-end machine learning pipelines. With *datastack*, we propose a framework that handles any data format as a byte stream on an interface level, and casts these byte streams to the desired data format within the dataset-specific components, such as the preprocessor or iterator. As a result, the framework itself is data format agnostic. Further, we implemented several higher-level iterator functions such as the stacking, joining, and splitting of iterators that can be arranged within a dependency graph, allowing complex iteration functionality. The sophisticated nature of outlier detector evaluation requires various datasets with different outlier types (e.g., observed, unobserved and noise) arranged in a complex evaluation pipeline. *Datastack* breaks this complexity down into higher level iterators.
- **MLgym**⁸: Machine learning research demands reproducible experiments, including tracking the implementation of the underlying experiment setup, the algorithm parameterization, and the performance scores. Most deep learning frameworks such as Tensorflow and Pytorch provide great flexibility in implementing, training and evaluating deep learning models, however they

⁷<https://github.com/lelnux/datastack>

⁸<https://github.com/mlgym/mlgym>

lack rudimentary support for reproducibility. Consequently, many machine learning researchers implement their code within a single jupyter notebook, resulting in a lack of test coverage, limited scalability, poor architectural design, and impaired readability. All of these shortcomings are partial causes for the insufficient reproducibility of machine learning research. Especially in outlier detection and open set recognition with their involved training and evaluation routines, there is a need for a well-designed framework handling the reproducibility of ML experiments. Therefore, we propose MLgym, a framework that tackles the aforementioned shortcomings by specifying the entire machine learning pipeline for training and evaluation within a single configuration file. Given the latest commit hash prior to an experiment, all experimental results can be reproduced using the respective config and commit hash. Furthermore, MLgym is a feature-rich framework that supports distributed model training/evaluation, grid search, (nested) cross-validation and common loss/metric functions.

Readers solely interested in the theoretical aspects of our work, can skip this chapter.

In Ch. 3, we turn towards supervised outlier detection, tackling the aforementioned challenges of supervised outlier detection. We propose two algorithms for supervised outlier detection based on semi-supervised outlier detectors and supervised autoencoders (SAEs) [1]. We explore how an adversarial loss function that incorporates outlier samples into the training objective significantly improves detection performance. Furthermore, we find that the reconstruction error score becomes highly predictive of the outlierness of a sample. Lastly, we find that the multi-task learning setup within SAE, combined with an adversarial loss function, leads to richer representations within the latent space that can be leveraged for downstream outlier classification.

In Ch. 4, we bridge the gap between supervised outlier detection and the related task of open set recognition. With DAE, we provide an extension to our proposed ATA method, which filters inliers and is capable of effectively rejecting outliers. As we will prove in Ch. 4, DAE bounds and minimizes the open space risk, thus limiting the false positive rate and learning a compact decision boundary around the inlier data. Over a range of experiments, we verify DAE's robustness superiority compared to multiple state-of-the-art OSR methods.

Having demonstrated the robustness benefits of DAE in Ch. 4, we apply these insights to multi-class classification in Ch. 5. By arranging DAE in a one-vs-rest ensemble, an architecture we refer to as *Informer*, we empirically show the tremendous robustness gains compared to traditional DNNs and other ensemble methods. Finally, we show that Informer splits aleatoric and epistemic uncertainty in a principled manner. The source of the two uncertainty types is fundamentally different and often calls for different actions, necessitating this property.

After proposing the three novel methods ATA, DAE, and Informer, we demonstrate, in Ch. 6, how these supervised autoencoder methods can be applied to practical use cases and lay out potential challenges. Firstly, we conduct a study on toxicity detection in online communication such as social networks. We find this task to be especially difficult for autoencoders, as learning a representation of normality is impractical due to the sheer size of the non-toxic class. However, our results show that learning a toxicity representation is feasible and generalizes better to unseen toxicity types in comparison to traditional deep learning methods. Secondly, we propose a document information extraction system for processing financial documents. We showcase how our methods seamlessly integrate with existing extraction algorithms at scale.

The thesis concludes with a summary and outlook in Ch. 7.

Design of the Experiment Environment

In this chapter, we outline the technical design of the experiment environment, we implemented to collect the empirical results for the following research chapters. The content of this chapter is not required to understand the theoretical underpinnings of our proposed methods and therefore can be skipped by readers only interested in the algorithmic aspects of this work. Nevertheless, this chapter is an important technical contribution to outlier detection and open-set recognition research. It exposes and solves various limitations of current deep learning research such as reproducibility and standardization of existing open-source frameworks. These limitations are enforced in outlier detection and open-set recognition which often require more sophisticated training and evaluation routines. For instance, in this setting, models are often evaluated w.r.t. different types and sets of outliers. Furthermore, outlier detection-specific training routines facilitating, e.g., over-/undersampling of a subset of classes and cost-sensitive learning, need to be implemented. These use cases and requirements are not met by current deep learning frameworks and force researchers to implement their own research environments from scratch. Moreover, test coverage and accurately designed architectures are mostly out of the scope of researchers, often resulting in low reproducibility or even faulty results in peer-reviewed articles.

To this end, in this chapter, we present the two open-source frameworks Datastack, a framework for end-to-end dataset preprocessing, and MLgym, a framework that dynamically specifies the entire training and evaluation workflows. Jointly, these frameworks provide the flexibility to create complex, reproducible workflows, meeting the special requirements of outlier detection and open-set recognition tasks.

More specifically, Datastack's key feature is dataset format agnosticism, which allows the framework to support any dataset format that can be represented as a byte stream. This is achieved by handling the datasets as byte streams on an interface level, irrespective of the underlying file format. Within the custom dataset implementation, these byte streams are decoded to the respective data formats for processing. Besides this feature, Datastack provides high-level iterator functionality and also allows implementing custom iterators that can be combined dynamically into complex dataset processing pipelines.

Complementing the dataset processing, MLgym allows researchers to perform reproducible deep learning research. Here, we leverage the inversion of control paradigm to propose a framework that, in contrast to most other deep learning frameworks, maintains full control over the training and evaluation cycle, while allowing its extension with custom components in a plug-and-play fashion. This design

approach significantly reduces the implementational overhead for researchers, as the framework already provides crucial functionality such as early stopping and checkpointing. Furthermore, the design allows the separation of the experiment setup (including Datastack’s dataset processing specification) from the code, thereby increasing the reproducibility of research results.

In the following sections of this chapter, we introduce the reader to Datastack and MLgym, and demonstrate how these frameworks benefit research in outlier detection and open-set recognition. Originally, this chapter is based on our publication [81] and open-source frameworks Datastack and MLgym. The development of the Datastack framework was lead by Max Lübbering, who contributed the central idea of achieving data format agnosticism via stream-level interfaces, designed the entire architecture, and contributed 99.5% of the lines of code¹. The Datastack publication [81] was first-authored by Max Lübbering, receiving feedback by the co-authors who had applied the framework to related research projects or participated in the frequent discussions.

Similarly, the MLgym framework development was lead by Max Lübbering, who contributed the principal ideas, resulting software architecture and the computation environment (including parallelization, GPU support, experiment parameterization, training, evaluation and logging). Max Lübbering also implemented the initial frontend for real-time experiment analysis, which students at Fraunhofer IAIS extended under his supervision.

2.1 Datastack: Unification of Heterogeneous Machine Learning Dataset Interfaces

In this section, we present our open-source Datastack framework which builds the foundation for the dataset processing pipelines in this work. While ML methods underwent a quantum leap in innovation in recent years, the ML DevOps tools have not matured at the same pace. Since experimental results heavily depend on data (pre-) processing, technical stagnation and lack of standardization have contributed to the frequently reported challenges of replicability and reproducibility [82–85] of ML research. From a data engineering perspective, datasets come in various formats such as Pandas DataFrames [86], Numpy Arrays [87], or CSV, shaping a heterogeneous format landscape. Many repositories, e.g., UCI Archive², Kaggle datasets³ and Microsoft Opendata⁴ host these raw datasets.

Integrating these datasets into ML pipelines usually comprises three recurring implementation steps:

1. Dataset retrieval from the hosting repository
2. Dataset preprocessing / data cleaning
3. Custom iterator implementation

As a result, data scientists and ML practitioners often reimplement these steps for every new dataset, leading to code duplication and poor architectural designs from the beginning that are susceptible to bugs. Dataset preprocessing / cleaning is especially unsatisfactory, as the implementation is often

¹<https://github.com/lelnux/datastack/graphs/contributors>

²<https://archive.ics.uci.edu/ml/index.php>

³<https://www.kaggle.com/datasets>

⁴<https://msropeodata.com/>

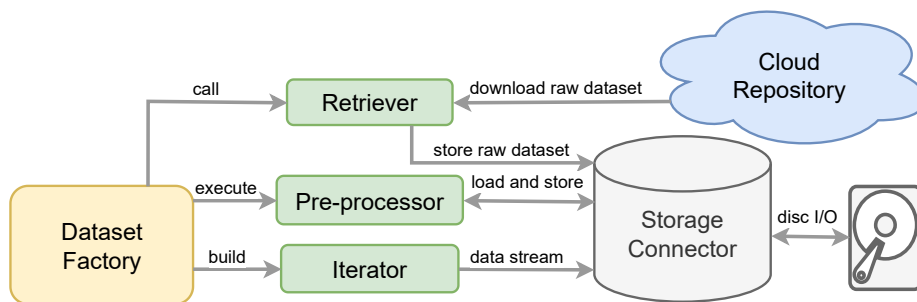


Figure 2.1: Datastack architecture: *Dataset Factory* commands *Retriever* and *Pre-processor* to download and prepare the dataset, respectively. Note that the preprocessor and iterator deal with arbitrary data formats internally. The interfaces and remaining components work on a byte stream level, thus decoupling the dataset-specific peculiarities from the framework.

spread over multiple scripts or Jupyter notebooks. In combination with possibly hardcoded, absolute paths, undocumented code, and various preprocessed dataset versions, productive collaboration within such projects becomes challenging.

There are multiple open-source projects such as HuggingFace Datasets⁵, TorchVision Datasets⁶, and TensorFlow Datasets⁷ that aim to alleviate the technical overhead of dataset integration. Usually, these libraries provide all the three steps of dataset integration mentioned above for an extensive number of datasets. While this is very convenient for quick prototyping and testing, extending these libraries to include new datasets is often nontrivial. This is because most libraries combine all three integration steps within a single class or file by importing a plethora of auxiliary functions. From a design perspective, this contradicts multiple programming principles [88], such as separation of concerns, single responsibility principle, and interface segregation principle, which limits the extensibility of these frameworks. Furthermore, these dataset repositories present the samples with framework-specific data structures, such as torch tensors within TorchVision. Therefore, custom datatype casting routines need to be implemented for cross-framework support. Nevertheless, these libraries are beneficial for quick prototyping when using the implemented datasets, because the design issues can be hidden behind a front-facing interface such as a facade [89].

For exploratory ML research, apart from overused benchmark datasets, we argue that there is a need for an architecturally well-designed integration framework for new datasets. To this end, we propose the Datastack⁸ framework, a simple yet effective open-source contribution that heads in this direction. Datastack decouples dataset retrieval, preprocessing, storage, and iteration, by introducing stream-based interfaces between the components mentioned above. Due to its interface design, Datastack becomes data format agnostic, since a byte stream can represent any data format, as shown in Fig. 2.1. Only the user-facing components such as the iterator or the preprocessor require ML framework / data format specific implementation.

The concept of (byte) streams is omnipresent in many data-driven computer science subfields. In machine learning, researchers have proposed various algorithms for stream data analysis [90, 91].

⁵<https://huggingface.co/docs/datasets/>

⁶<https://pytorch.org/vision/stable/datasets.html>

⁷<https://www.tensorflow.org/datasets>

⁸<https://github.com/lelinux/datastack>

In high-performance computing, stream processing pipelines are an ongoing field of research with Flink [92], Storm [93], and Spark [94] being highly-scalable representative frameworks for stream processing. These frameworks provide the infrastructure to process data regardless of the initial input format, making these frameworks data-format agnostic.

Interestingly, to the best of our knowledge, these concepts have never been applied to dataset (pre-) processing within ML pipelines. With Datastack, we demonstrate that the aforementioned deficiencies in ML dataset processing/integration can be solved by introducing streams on an interface level.

2.1.1 Design Choices

The vast number of different file formats for datasets requires dataset integration frameworks to support arbitrary file and data formats. We achieve this goal by treating dataset objects as byte streams⁹, which can be thought of as a seekable data block stored either on disk or in memory. Since every major data container object provides a file-like object¹⁰ interface, they automatically support byte streams. This setup gives us great flexibility and consistency within the framework design since we only need to pass byte streams between modules irrespective of the internal data format.

Every dataset implementation has an iterator, which seamlessly builds a generic interface for training routines of the most prominent ML frameworks such as scikit-learn [95], TensorFlow [96], or PyTorch [97]. We provide higher-level functionality such as dataset splitting, target filtering, target encoding, shuffling, and in-memory loading, all of which are achieved by stacking, splitting, or joining iterators, as shown in Fig. 2.2. A higher-level iterator has a read-only view of the underlying iterator, which allows for a minimal memory footprint and negligible runtime complexity, due to direct indexing. In practice, ML engineers often reimplement (or worse, copy and paste) these data processing routines with every new dataset, resulting in duplicated code that is susceptible to common ML bugs like information leakage [98]. Datastack's dataset processing routines are thoroughly tested and new routines can be derived from the lower-level stacking, joining, and splitting routines. Additionally, domain-specific operations such as augmentation in computer vision or text embedding in natural language processing can be implemented.

For our outlier detection experiments, we heavily rely on the joining, stacking and splitting operations to create different iterators, e.g., with distinct types of outliers for training and evaluation. Considering the outlier types in different iterators, allows the investigation of their influence in isolation. Further, the file-format agnosticism is leveraged throughout the experiments, as the raw datasets come in various formats.

2.1.2 Implementation

At Datastack's core, there is the *Storage Connector* which provides a generic data streaming I/O interface for storing raw and preprocessed data, as shown in Fig. 2.1. We thereby support various storage solutions ranging from in-memory databases and local file storage to distributed storage solutions. In its minimal deployment version, Datastack comes with a persistent key-value storage. This on-disc storage provides the saving and retrieving functionality for any byte stream that is

⁹<https://docs.python.org/3/library/io.html#module-io>

¹⁰<https://docs.python.org/3/glossary.html#term-file-object>

2.1 Datastack: Unification of Heterogeneous Machine Learning Dataset Interfaces

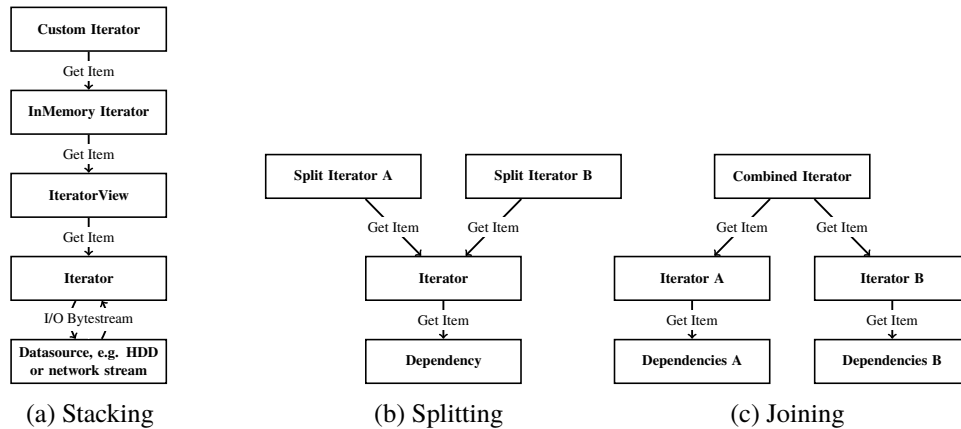


Figure 2.2: Combination of dataset iterators via stacking, splitting, and joining operations: Similar to table views in databases, each higher-level iterator represents a view on its underlying iterator. The stacking, splitting, and joining operations build the foundation for more sophisticated iterators with functionality such as in-memory loading or label mapping.

identified by a unique key. For enterprise-level solutions, the same interface already integrates with Amazon S3¹¹ and Minio¹².

The components within Datastack interact as follows: The *Dataset Factory* commands the *Retriever* to download the raw dataset from a given cloud repository or local file path. After retrieval, the *Preprocessor* receives the raw dataset as a byte stream from the *Storage Connector*. It then performs data cleaning, shapes the raw data into a usable format for the machine learning algorithm, and saves the preprocessed/cleaned data stream via the *Storage Connector*. Here, the dataset retrieval and preprocessing steps only need to be performed once, as the preprocessed data stream can be reused. Finally, the *Iterator* retrieves the preprocessed data as a byte stream from the *Storage Connector* and provides the iteration functionality. Note that the *Preprocessor* and *Iterator* internally use custom data representation formats (e.g., Pandas DataFrames, or CSV) for the datasets, however, on an interface level between modules (e.g., between the *Preprocessor* and the *Storage Connector*), the representation is always handled as a byte stream. Thus, the framework itself is agnostic of any specific file / data formats, while supporting arbitrary data formats within the custom preprocessor and iterator implementations.

Furthermore, the iterator design improves the flexibility by allowing the, e.g., stacking, splitting, and joining of iterators, as illustrated in Fig. 2.2. Implementation-wise this is similar to the table view concept within databases, as an iterator view holds a list of indices that index the underlying iterator elements. This has a time complexity of $O(1)$. The splitting/shuffling functionality is implemented by shuffling the indices list or considering only a subset of indices within the iterator view, respectively. Here, a combined dataset iterator represents a conjunction of views on the respective underlying iterators. This view concept allows for the derivation of further high-level postprocessing routines like, e.g., target label mapping or label filtering.

Given enough system resources, the final iterator can be loaded into memory, since this iterator can be considered to be immutable. Thus, the runtime complexity of indexing is improved from $O(n)$ to

¹¹<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html>

¹²<https://github.com/minio/minio>

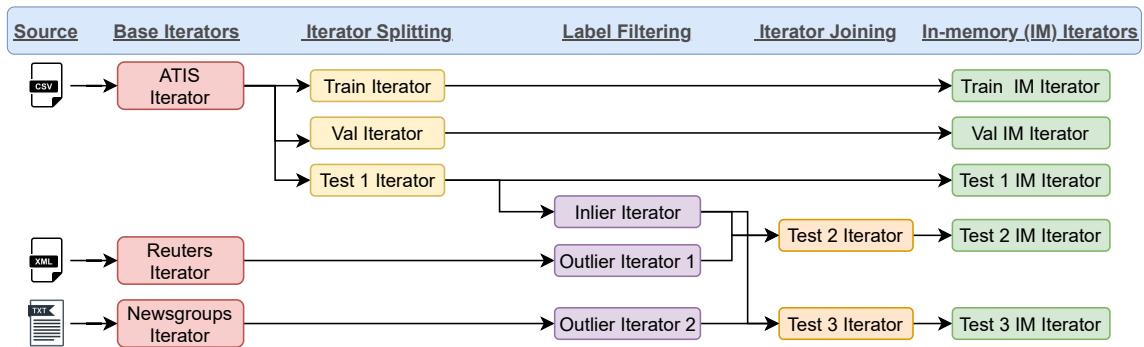


Figure 2.3: Use case example on textual outlier detection [22, 99]: Splitting, stacking, joining, label filtering, and in-memory loading is used to represent the full data processing pipeline. The final in-memory iterators are used for training, validation, and evaluation of ML models.

$O(1)$ where n is the stack size. Additionally, the postprocessing routines such as label mapping are only invoked once. The generic iterator interface can be easily integrated into state-of-the-art ML frameworks such as TensorFlow and PyTorch with their dataloader implementation, as we will show in Sec. 2.2 for MLgym.

2.1.3 Use case: Dataset Processing Pipeline for Outlier Detection

As described in Sec. 1.2, the goal of outlier detection is to distinguish observed inlier samples from outlier samples that are derived from an outlier generating process. Often, models are trained solely on the inlier data, and at testing time, are evaluated against outliers. In practice, the outliers are often sampled from contextually unrelated datasets [99, 100]. In Fig. 2.3, we showcase how Datastack can support deriving the different iterators for training and evaluation based on our outlier detection research [22, 99]. In this experiment setup, we train an algorithm on the ATIS dataset, and at testing time, investigate the algorithm’s robustness to outliers originating from the Reuters and Newsgroups datasets. All datasets come in different file formats (i.e., CSV, txt, and XML), which Datastack handles on a byte stream level. Once the datasets are preprocessed/cleaned by the dataset-specific preprocessor, the factories build the respective base iterators.

The base iterators are used to create the final iterators by applying routines such as stacking or joining. First, the ATIS iterator is split into train, val, and test iterators. These iterators can be directly used for training, validation, and evaluation, which is why they are loaded into memory at the end, without further processing. From the Reuters and Newsgroups iterators, we filter a subset of the labels to create the outlier iterators. For model evaluation, we need to combine the outlier iterators with the inliers from the test 1 iterator, which results in the test 2 and test 3 iterators. Finally, these two iterators are loaded into memory. In conclusion, this use case highlights the advantages of Datastack: Multiple datasets with different raw data formats can be easily processed and combined into a complex, bug-tolerant pipeline using simple stacking, joining, and splitting functionality.

2.1.4 Conclusion and Outlook

With Datastack, we provide a generic, open-source framework for ML datasets that can be seamlessly integrated into existing ML frameworks. While datasets are prevalent in various formats, the framework

internally treats them as binary streams, making the framework data-format agnostic. Furthermore, it is designed to have well-defined interfaces and reusable components, allowing the implementation of new datasets with minimal effort. Datastack has extensive high-level iterator functions such as dataset splitting and shuffling inspired by the table view concept from the databases domain. We showcased how this functionality can be utilized in the experiment setup for outlier detection.

For future work, we are aiming to showcase Datastack within a large-scale dataset repository [101] for outlier detection datasets. From an industrial point of view, showcasing Datastack’s support for enterprise-level storage solutions would be another essential direction. Since Datastack is entirely open-source, we invite other researchers to use its dataset processing functionality and finally would like to establish a community-driven development. In the following section, we introduce the MLgym framework and explain how Datastack has been integrated.

2.2 MLgym: Architectural Proposal for Reproducible, Standardized Deep Learning Research

While the results of SOTA models are generally collected in benchmark rankings in various places^{13,14,15}, their reproducibility often poses a challenge with reasons ranging from selective reporting (cherry picking) and lack of code availability, to poor experimental design. A 2016 study [102] states that among 1,576 surveyed researchers, 52% agreed that there is an ongoing reproducibility crisis, and even 70% failed to reproduce experiments.

Looking specifically at the field of machine learning, reproducibility is an even more significant challenge, considering that, among the papers published at top machine learning conferences, only about 6% include code, and 30% provide test data [84, 85]. Furthermore, the authors of [84] demonstrated, as part of an extensive reproducibility study of 400 papers published at premier ML conferences, that on average, only 24% of the variables were documented to achieve full replicability of the paper results.

With the advent of reproducibility challenges [103, 104] hosted by premier ML conferences, which encourage researchers to replicate the claims of research papers, there is an even greater need for standardized tooling that enables fast-paced experimental reproduction. While ML conferences (e.g., NeurIPS, ICML, ICLR) and researchers have defined checklists¹⁶ for improved reproducibility [105, 106], they only define the criteria, not how to fulfill them in a standardized fashion. As a result, researchers often have to invest significant amounts of time in digging through hundreds of lines of possibly poorly documented code within publicly hosted repositories to understand the underlying infrastructure before rerunning the experiments and reproducing a paper’s claims. We argue that with standardized infrastructure tools, researchers can focus solely on verifying the implementation of the concrete methodology regarding the proposed model and its evaluation.

While socioeconomic and political factors among others are impeding reproducibility [107–109], we focus on the technical challenges in this work [82, 83]. In particular, we follow the argumentation of [82, 110], who argue that replicability/repeatability is the first step towards reproducibility. To this

¹³<https://nlpprogress.com/>

¹⁴<https://paperswithcode.com/sota>

¹⁵<https://sotabench.com/>

¹⁶<https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf>

end, we propose MLgym¹⁷, an open-source deep learning framework based on PyTorch [97], which provides a vital cornerstone for low-effort replicability, and the tools to reproduce insights gained over a proposed method.

Already in 2013, [111] determined ten rules to achieve reproducible computational research in their highly cited paper, of which six rules directly apply to machine learning frameworks:

- Rule 1: *“For every result, keep track of how it was produced”*
- Rule 2: *“Avoid manual data manipulation steps”*
- Rule 4: *“Version control all custom scripts”*
- Rule 5: *“Record all intermediate results, when possible in standardized formats”*
- Rule 6: *“For analyses that include randomness, note underlying random seeds”*
- Rule 10: *“Provide public access to scripts, runs, and results”*

Now, the pressing question is why all existing deep learning frameworks fail to implement most of these rules. We argue that the reasons lie within the architectural design of these frameworks (e.g., PyTorch [97] and TensorFlow [96]), which merely provide a set of functions and force the researcher to implement the training and evaluation pipeline from scratch. On one hand, this architectural design provides maximum flexibility and allows for fast prototyping, which is beneficial for conducting ML research. On the other, the pipelines are out of scope of the respective ML framework, limiting the possibilities for internally implementing the reproducibility features to comply with any of the rules outlined above. Consequently, almost any implementation regarding reproducibility is left for the researcher to implement, a cumbersome and error-prone task, easily leading to wrong results and conclusions in research papers.

To this end we propose the open-source framework MLgym, which is tailor-made for reproducible deep learning research, and alleviates the trade-off between the implementation’s flexibility and the result’s reproducibility. Following the inversion of control (IoC) paradigm [112], MLgym implements the aforementioned pipelines internally and allows the researcher to extend the functionality by registering custom component classes (e.g., models, evaluators, trainers). Therefore, the framework maintains complete control over the training and evaluation process, allowing for arbitrary experiment tracking. The design yields two additional advantages: Firstly, it allows the user to specify the entire training and evaluation pipeline as a dependency graph within a configuration file. This configuration file can be instantiated dynamically via dependency injection during the initialization of MLgym, already matching the requirements in rules 1, 2, 4, and 6. Secondly, MLgym comes with distributed logging functionality based on event sourcing [113], using websocket streams to address rules 5 and 10. In the minimal deployment, we use a centralized websocket server to save the incoming messages regarding training progress and evaluation stats within a persistent event store. All the collected experiment data is accessible via the websocket API and a RESTful API, and can be inspected by the included visualization frontend MLboard.

The following sections are organized as follows: In the next section, we discuss related deep learning frameworks and explain how and to what extent they support reproducibility. In Sec. 2.2.2, we introduce the architectural design and discuss how it benefits reproducibility. We then introduce

¹⁷<https://github.com/mlgym/mlgym/>

the entire system in Sec. 2.2.3, followed by an exemplary use case showcasing the practical benefits of MLgym in Sec. 2.2.4. We outline the possible future directions beyond reproducibility, such as Green AI [114] in Sec. 2.2.5 and finally, formulate the conclusion in Sec. 2.2.6.

2.2.1 Related Work

As outlined in the previous section, increasing the overall amount of reproducible research results is a crucial yet underexplored and underdeveloped endeavour in the field of ML research [84]. This section demonstrates the current state-of-the-art reproducibility research from a technical point of view, by showcasing the tools currently used in deep learning research.

Today, TensorFlow [96] and PyTorch [97] are the two most prominent deep learning framework representatives with which researchers implement their approaches. These frameworks provide the functionality to develop arbitrary deep learning architectures, the algorithms to perform backpropagation [3], multiple state-of-the-art optimizers, and various utility functions. However, both frameworks compromise implementational efforts for flexibility, by leaving the model’s training and evaluation routines to the researchers.

As the two frameworks can be regarded as a minimal backbone, researchers either implement the infrastructure routines for model training, evaluation and selection, or resort to higher-level frameworks. Various web-based tools have been developed for logging ML experiments, such as Tensorboard¹⁸, Weights&Biases [115], and Sacred [116]. Tensorboard provides simplistic experiment tracking functionality w.r.t. metrics and loss developments. Weights&Biases and Sacred additionally track hyperparameter configurations, allowing us to inspect the influence of certain hyperparameter configurations on the model performance. All three logging tools reduce the boilerplate code w.r.t. logging and the susceptibility to bugs to some extent, however, their impact on reproducibility is limited as the user still has to implement the entire training workflow and collect the information (e.g., metric/loss values, hyperparameter combination, GPU utilization) to be passed to the frameworks’ logging routines.

2.2.2 Architectural Overview

As anticipated, middleware solutions have gained traction in the research community in recent years. Most prominently, PyTorch Lightning [117] maps out the entire training workflow, including high-level functionality such as early stopping and warm starts. Architecturally, Lightning expects the model to implement a fixed set of Lightning-specific method hooks to instantiate optimizers and perform data preprocessing, training, and evaluation steps. Lightning’s trainer class implements the training loop and automatically calls the respective pre-defined hooks, reducing the amount of boilerplate code, however, this design also expresses some pitfalls. Firstly, it mixes the various concerns of model definition, training, and evaluation that, as we argue, should be decoupled instead. Secondly, it generally facilitates inheritance to override the hooks and abstract implementations, not adhering to the composition over inheritance paradigm. Both design issues lead to bloated interfaces and a static training pipeline. Since custom functionality is injected via inheritance, the concrete implementation comprises procedural code that is not interpretable by the framework, despite following the inversion of control paradigm to some extent. Furthermore, for model selection, cross-validation or grid

¹⁸<https://www.tensorflow.org/tensorboard>

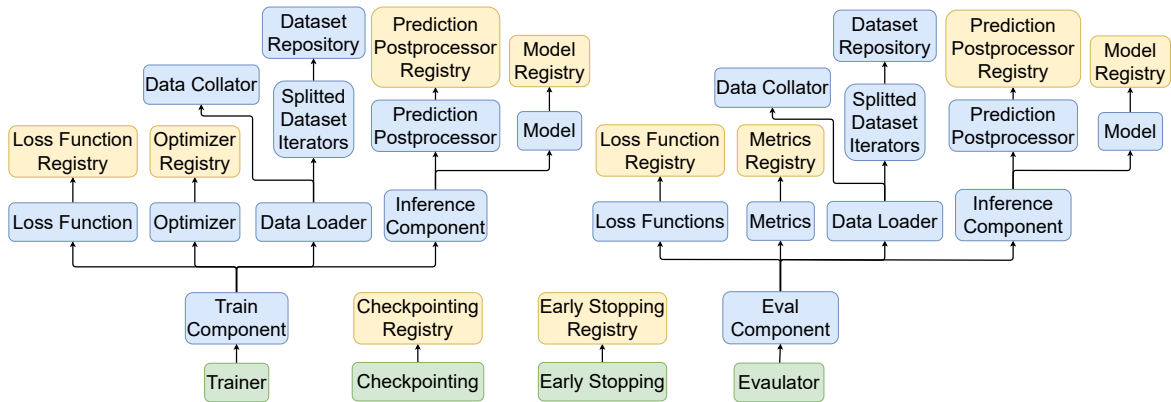


Figure 2.4: The YAML configuration file specifies the pipeline components, their dependencies, parameterization and the hyperparameter search. Here, the dependency graph of an exemplary deep learning configuration is showcased. To run the training and evaluation pipeline, we instantiate the root components trainer, checkpointing, early stopping and evaluator by passing the dependent components to the constructors via dependency injection. In most cases, a component-specific registry on the lowest dependency level instantiates the respective component. Even though the train and eval components are partially instantiated with the same components, we keep two separate instances to prevent side effects during training/evaluation.

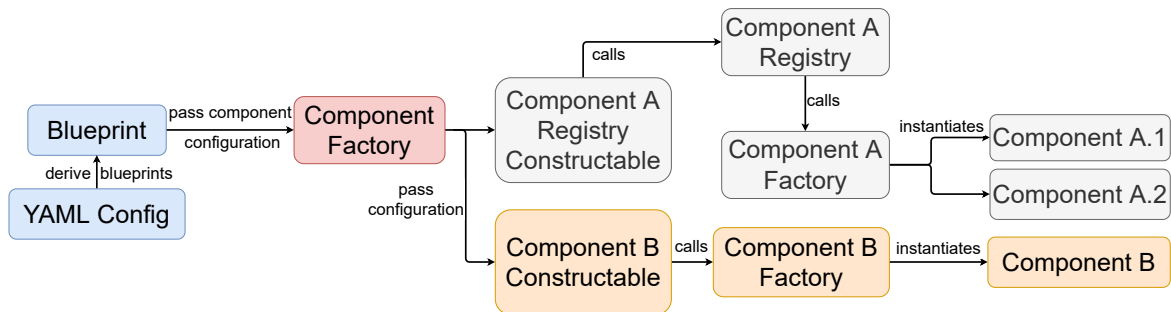


Figure 2.5: Pipeline component instantiation in blueprint via the component factory: A single experiment config from the YAML config is passed to the blueprint, a data class describing the entire pipeline setup. The blueprint commands the component factory, with its internal (custom) component registry, to instantiate the components defined in the dependency graph within the configuration. We distinguish two types of components here. Firstly, some components can be grouped (e.g., F1 score and accuracy) and do not require individual registration. These components are subsumed within a registry (see Component A Registry which registers Component A.1 and Component A.2). Secondly, standalone pipeline components (e.g., train component) have a dedicated constructable that only instantiates this particular component (see Component B).

searches, researchers either have to implement the functionality from scratch or combine Lightning with hyperparameter optimization [118, 119], and visualization frameworks [115].

MLflow [120], on the other hand, with its tracking component, directly interfaces with PyTorch Lightning and other ML libraries by monkey patching the logging within the training routine, or expecting a custom training routine that explicitly calls the logging routine, similar to the logging frameworks mentioned previously.

In conclusion, various high-level ML frameworks provide auxiliary functionality to reduce boilerplate code. Nevertheless, it is mainly the researcher’s obligation to implement the middleware code that

wires up all the components, such that the research environment adheres to rules 1, 2, 4, 5, 6 and 10.

With MLgym, we aim to alleviate the patchwork of various ML tools/frameworks by providing a single, standalone solution that dynamically maps out the entire training and evaluation pipeline internally. We define four principal requirements towards MLgym that distinguish the framework from any existing solution concerning features, architecture, and reproducibility:

- The training and evaluation pipeline should be defined in an end-to-end fashion comprising dataset preprocessing, model definition, model training, and model evaluation.
- For full reproducibility, the experimental design setup in its entirety should be defined separately from the code and be subject to version control.
- Every pipeline component should be extensible or replaceable without modification of the existing code.
- The framework should implement tested, high-level functionality such as grid search, (nested) cross validation [121], early stopping, warm starts, results visualization, multiprocessing, data preprocessing, and distributed logging.

As outlined previously, these requirements cannot be matched by any of the introduced frameworks without significant architectural changes, making our architectural design a novelty in the DL domain. In MLgym, we employ the inversion of control principle by implementing all components within the code and defining the pipeline (i.e., the assembly of the components) by including the component parameterization within an external config. Decoupling the pipeline configuration from the framework implementation has multiple practical advantages: 1) Different experiments can be run without having to change any code due to the dynamic pipeline assembly, leading to a well-maintainable code base and history of experiments. 2) Upon publication, a research repository only needs to provide the pipeline configuration and the custom components, allowing reproduction of the experimental results in a fast and standardized manner. 3) Each experiment defined in the config is self-contained and serialized, simplifying deployment in multiprocessing and multi-node environments.

Technically, the YAML configuration file provides the pipeline as a dependency graph and optionally different parameterizations for hyperparameter optimization/grid search, as illustrated in Fig. 2.4 and Listing 1. At runtime, the configuration is split into individual experiment configs based on the different hyperparameter combinations. Each experiment config is stored within a blueprint data class, comprising all the information to build the pipeline. As indicated in Fig. 2.5, the component factory traverses the dependency graph to instantiate the root components' respective dependencies and parameterizations (dependency injection). The blueprint class is implemented by the researcher and can register new custom components, which would otherwise be out of the scope of MLgym. Furthermore, the blueprint is serializable, allowing the job to be run in distributed, multi-node clusters out of the box.

There are two options to implement a new component, as shown in Fig. 2.5. In the first option, grouped components such as loss functions or metric functions are subsumed within a component registry to prevent having to register every group item with the component factory individually. The component registry instantiates the correct component based on the provided config, following the strategy pattern. In the second option, the remaining components are directly registered with the component factory.

```
1 splitted_dataset_iterators:
2   component_type_key: SPLITTED_DATASET_ITERATORS
3   variant_key: RANDOM
4   requirements:
5     - name: iterators
6       component_name: dataset_iterators
7       subscription:
8         - train
9         - test
10  config:
11    split_configs:
12      train:
13        train: 0.7
14        val: 0.3
15      seed: 2
16
17 data_collator:
18   component_type_key: DATA_COLLATOR
19   variant_key: DEFAULT
20   config: [...]
21
22 data_loaders:
23   component_type_key: DATA_LOADER
24   variant_key: DEFAULT
25   requirements:
26     - name: iterators
27       component_name: splitted_dataset_iterators
28       subscription: [train, val, test]
29     - name: data_collator
30       component_name: data_collator
31   config:
32     batch_size:
33       sweep: absolute
34       values: [10, 20, 50]
35     seeds: [0, 1, 2]
```

Listing 1: A configuration excerpt showcasing the hyperparameter optimization (see `batch_size` key) and the data loader’s dependency on data collator and splitted dataset iterators (see `requirements` key). While the excerpt illustrates only the data processing pipeline, the entire ML pipeline follows the same principles, as shown in Listing 2 in the Appendix. Note that, this level of standardization already provides a solution to rule 1, 2, and 6, as well as partially to rules 4, 5, and 10.

Internally, the component factory acts as a registry [122] that maintains a mapping from the (component key, variant key) tuple to the component constructable. The variant key allows specifying different implementations for the same component type. This is a crucial feature as it allows us to build pipelines with custom components dynamically without any code changes, following the open-closed principle [123]. Furthermore, the variant key can provide the means for the components’ deprecation management. The component constructable can be regarded as a meta-factory that adds the respective component to the component factory, and thereby, to the framework’s scope. It is noteworthy that the framework itself is always programmed against component-specific interfaces. These are formulated as contracts that need to be implemented by the respective (custom) components,

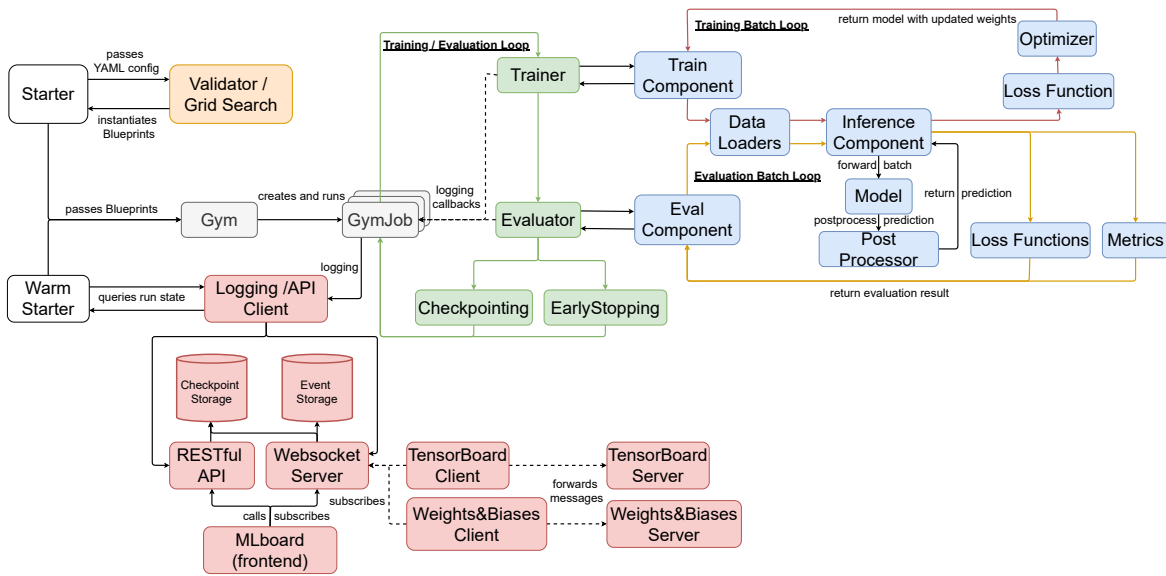


Figure 2.6: Illustration of MLgym system design: The two entry points, *starter* and *warm starter*, forward the blueprints (specification of the experiments) to the multiprocessing gym for execution. As specified in Fig. 2.4, the root and dependency components, highlighted in green and blue, respectively, jointly resemble the dynamically assembled training/evaluation pipeline. The logging environment based on event sourcing and streaming is highlighted in red and supports custom clients, e.g., TensorBoard or Weights&Biases. With the entry points and logging environment, MLgym fulfills rules 5 and 10.

encouraging modularity.

To further increase the reproducibility of model training and evaluation, we adopt event sourcing [113] to track various types of state changes during model training and evaluation, as atomic and timestamped event messages. The message history allows the user to replay and analyse the entire experiment run a posteriori, which we leverage in the implementation of our frontend MLboard.

The architectural design outlined in this section provided the foundation for the overall system, discussed in the following section.

2.2.3 System Design

The system design of MLgym can be separated into the five modules shown in Fig. 2.6, namely the two entry points *starter* and *warm starter*, the validators (orange), the multiprocessing environment (grey), the logging (red), and the dynamic pipeline (green and blue) that can be divided into the two training and evaluation sub-pipelines.

The two entry points collect the blueprints with the experiment configs and pass them to the gym. In case of a new experiment run, the starter passes the YAML config to the validator, which breaks it down into individual experiment configs. MLgym supports (nested) cross-validation and grid search out of the box. For any other validation method, a researcher can provide a custom validation method by implementing the validator interface. In case of continuing a stopped experiment run, the warm starter queries the RESTful API to retrieve the experiment configurations and latest checkpoints, and forwards the respective blueprints to the gym.

The gym can be regarded as the central computation unit that provides a multiprocessing environment

to run the experiments encapsulated in gym jobs in parallel. Technically, the gym constructs a computation job from the serializable blueprint within the respective process, without the need for intra-process communication and complex data exchange. A gym job directly calls the root components *trainer*, *evaluator*, *checkpointing*, and *early stopping* within its training/evaluation loop. For each batch within the training batch loop, the inference component comprising the model and postprocessor predicts the batch and forwards the prediction to the loss function. After backpropagating through the model, the optimizer updates the weights. Similarly, the evaluation batch loop calculates the losses and metrics for each batch and returns an evaluation batch result object.

Note how this design decouples training and evaluation functionality from the model. The model merely acts as a data class, storing the model weights/architecture and provides a rudimentary forward function. Due to the decoupling and application of IoC, we support dynamically assemblable training and evaluation strategies without having to change the model code, as would be the case for DL frameworks such as PyTorch Lightning. Furthermore, every pipeline component in MLgym implements a respective interface which allows the user to add and replace components with custom implementations in a plug-and-play fashion.

MLgym utilizes Datastack (see Sec. 2.1) to provide extensive functionality for data (pre-) processing, including splitting, merging, shuffling, lazy loading and in-memory loading of datasets. With its possibility to build complex dataset pipelines and its dataset format agnosticism, Datastack's design nicely matches the component-based pipeline structure within MLgym. As a result, arbitrary data preprocessing pipelines can be realized within MLgym by combining generic and custom dataset iterator components.

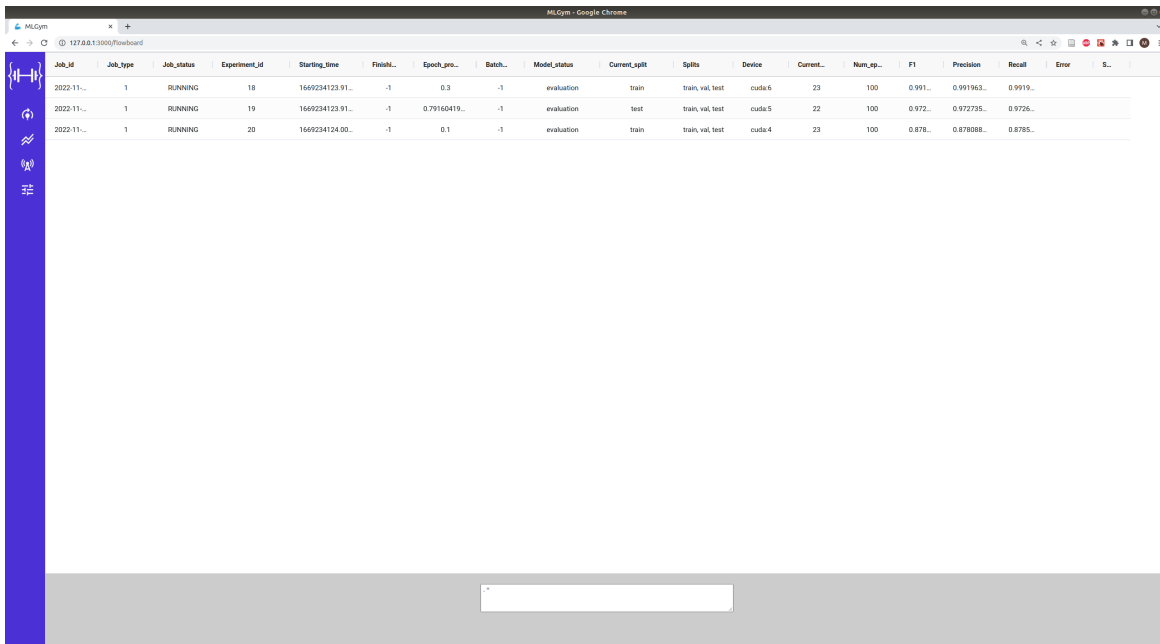
The trainer and evaluator provide callback functions to the dependent components to update progress and performance results. The gym job logs these messages and checkpoints via the websocket streaming API. The backend utilizes event sourcing and an event storage to persistently track the messages, allowing the reconstruction of the entire training and evaluation procedure within the MLboard frontend. Note that the serializable blueprints and websocket logging infrastructure allows for a distributed system with, e.g., model training/evaluation performed on a high-performance computation (HPC) cluster, and results being logged on a dedicated storage server. The MLboard frontend and other remote clients can subscribe to the event-sourced messages for visualization or forwarding purposes. MLboard is showcased in Fig. 2.7.

2.2.4 Representative Research Use Case

In research, the effectiveness of any novel deep learning architecture is generally verified on multiple benchmark datasets and benchmarked against baselines to put the proposal's performance into perspective. MLgym is tailor-made for this representative research use case, and we outline the typical steps in this section:

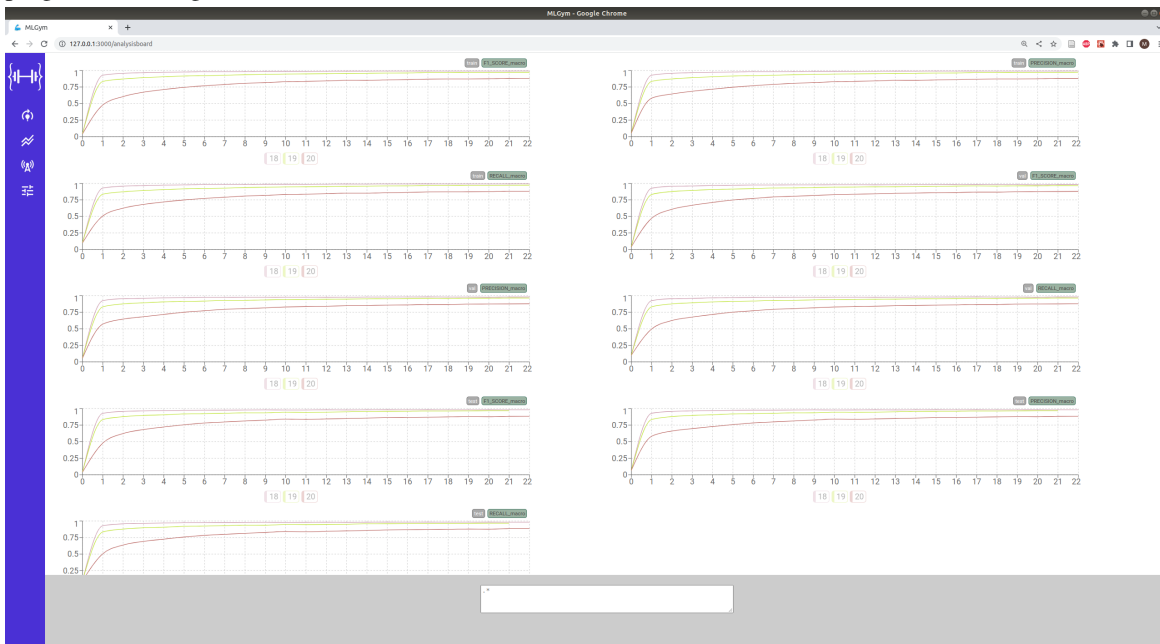
1. Implementation of the data preprocessing components for each benchmark dataset based on Datastack (see rule 2 in Sec. 2.2)
2. Implementation of the proposed DL architecture and the corresponding baselines
3. Implementation of a custom blueprint that registers the new components (i.e., dataset repository with the benchmark datasets, model registry with architecture proposal and baselines)

2.2 MLgym: Architectural Proposal for Reproducible, Standardized Deep Learning Research



Job_ID	Job_type	Job_status	Experiment_ID	Starting_time	Fields	Epoch_pre	Batch	Model_status	Current_split	Splits	Device	Current...	Num_ep...	F1	Precision	Recall	Error	S...
2022-11-...	1	RUNNING	18	1669234123.91...	-1	0.3	-1	evaluation	train	train, val, test	cuda:6	23	100	0.991...	0.991963...	0.9919...		
2022-11-...	1	RUNNING	19	1669234123.91...	-1	0.79160419...	-1	evaluation	test	train, val, test	cuda:5	22	100	0.972...	0.972735...	0.9726...		
2022-11-...	1	RUNNING	20	1669234124.00...	-1	0.1	-1	evaluation	train	train, val, test	cuda:4	23	100	0.878...	0.878088...	0.8785...		

(a) The flow tab renders a data table, aggregating crucial information for each experiment in a single table row. The tab provides information about the job status, hyperparameterization, training progress and evaluation progress including metric/loss scores.



(b) The analysis tab renders the metrics and losses concerning the different dataset splits for each experiment as a line chart.

Figure 2.7: MLboard frontend: a React application that visualizes the training and evaluation progress by subscribing to the websocket API and REST API.

4. For each dataset and model, we define a YAML config outlining the training/evaluation pipeline including hyperparameter sweeps, checkpointing, and the early stopping strategy.
5. We execute the pipelines and capture the results in a centralized event storage. If the pipeline crashes at any point (e.g., due to GPU memory overflow or unrelated server issues), we can analyse the issue as we track the entire message history via event sourcing and resume the training from the most recent checkpoint via the warm start entry point.
6. During the training, we can start to analyze the performance of the trained and currently training models, and can save/share the analysis with our peers.
7. After publication of the results / research paper, the code, experiment configs and analysis configs (MLboard) are made publicly available within a version controlled repository.
8. Other researchers can clone the repository, install the package dependencies, rerun the experiments, and verify the correctness of the communicated results. Furthermore, researchers can also integrate new benchmarks and baselines to the repository or, vice versa, integrate the datasets and models into their own repositories.

As outlined in the eight steps above, MLgym yields reproducibility gains unmatched by any other (meta) deep learning framework, which usually require significant implementational efforts from the researchers that are plainly impractical. With MLgym, these efforts are minimized to such an extent that reproducibility becomes feasible.

2.2.5 Quo vadis?

Despite the striking reproducible benefits of MLgym, various directions are still worth exploring and implementing. Firstly, the gym component utilizes multiprocessing to distribute the jobs on different GPUs within a single node. With the continuously increasing number of weights in deep learning architectures, the size of models (e.g., large language models [5]) sometimes already exceed the GPU memory on a single node. Therefore, future work needs to be invested in distributing model training across nodes, e.g., via deepspeed [124].

Another technical direction is to add further frontend functionality, as the current version of MLboard is rather rudimentary. The current version provides a live feed on the training progress and metric/loss developments for each experiment with the respective parameterization, as shown in Fig. 2.7. Consequently, there are still various directions for automated higher-level analysis and comparison. For instance, cross validation, nested cross validation, and grid searches have different purposes [121] and require different evaluation routines. Adding this analysis functionality to MLboard would assist the researcher in the analysis of the results.

An often overlooked advantage of reproducibility is its applicability to Green AI [125], since already-run experiments can be identified by the experiment config and skipped in the future. Further, MLgym can lower the carbon footprint of model training due to its distributed design. The energy-intensive training and evaluation can be performed on an HPC cluster running with green energy or scheduled for times and locations with an electricity surplus. The gym instance on the HPC cluster sends the checkpoints and results to a remote logging environment for inspection by the researcher. Another direction is estimating the carbon footprint of the training, evaluation and inference of the model, e.g., in terms of the number of floating point operations as a surrogate metric [114]. Including

efficiency metrics puts performance gains into perspective, rendering some models' improvements debatable [105, 126].

2.2.6 Conclusion and Outlook

Lack of results reproducibility is a critical pain point for machine learning research that has been called out as a reproducibility crisis by many researchers. In this section, we focused on the technical aspects impeding reproducibility. We identified the architectural, procedural design of deep learning frameworks as a core obstacle that forces researchers to implement boilerplate code to achieve high reproducibility.

To this end, we showcased MLgym, a feature-rich, PyTorch-based deep learning framework that focuses on reproducibility. MLgym applies the inversion of control paradigm, providing a generic framework for model training and evaluation by allowing researchers to implement and register their custom components. As a result, we can specify the pipeline in a single config file, separating the experimental setup completely from the code and dynamically assemble the training and evaluation pipeline at runtime. Furthermore, MLgym comes with crucial high-level features such as early stopping, warm starts, checkpointing strategies, and standardized, event-sourcing-based logging that tremendously cut down boilerplate code.

Since we are confident that MLgym's architectural design and various features can significantly benefit the reproducibility and accessibility of research results, we decided to open-source MLgym¹⁹ and strive for a community bringing in novel ideas and directions.

2.3 Conclusion

The complexity of outlier detection and open-set recognition experiment setups, and the lack of infrastructure tools in deep learning, has encouraged us to implement our two open-source contributions Datastack and MLgym. We engineered both frameworks to maximize reproducibility without compromising flexibility of deep learning experiments. In Datastack, we specify the entire data pipeline, comprising dataset preprocessing, splitting, merging and various other operations as stacked iterators. As the interfaces exchange data on the most generic data format, i.e., binary streams, we can define the entire data pipeline in an end-to-end fashion, irrespective of the underlying data format. Similarly, MLgym specifies the entire training and evaluation pipeline of deep learning experiments separated from the infrastructure and model implementation, increasing the reproducibility significantly. The generic interfaces within MLgym allow the replacement of any machine learning component within the framework with custom implementations at runtime. The two frameworks complement each other, jointly increasing the reproducibility of experiments without any compromises in flexibility. Datastack and MLgym provide the infrastructure for all our experiments.

¹⁹<https://github.com/mlgym/mlgym/>

Supervised Outlier Detection with Deep Neural Networks

In this chapter, we explore the applicability of autoencoders for supervised outlier detection. As explained in Sec. 1.2, unsupervised outlier detection methods are applied with the intention of filtering data that is out of the ordinary and passed to experts for review. Therefore, unsupervised methods can be an effective approach when outliers cannot be further specified. However, in most settings, we are aware of what type of outliers we are interested in. In this case, it is generally advised to leverage context dependent outliers at training time to integrate a contextual outlierness signal into the model, improving the effectiveness of the model at deployment time and rendering expert review redundant. For instance, to train a camera-based home alarm system, it can be beneficial to include video footage of actual intruders, so the system is not set off by your neighbor’s straying dog in the middle of the night.

To this end, we explore two supervised outlier detection methods. Firstly, building upon the idea of semi-supervised, one-class autoencoders (OCA), we propose a new supervised version, namely, an *adversarially trained autoencoder* (ATA), which maximizes/minimizes the reconstruction error of outliers/inliers, respectively. This end-to-end outlier detection algorithm enriches the reconstruction error as an outlier score with outlier information, effectively separating inlier and application-specific outliers.

Secondly, we explore the direction of supervised outlier detection w.r.t. multi-task learning (MTL) by jointly training an autoencoder on the reconstruction objective, and an outlier classification objective based on the autoencoder’s latent state. Our proposed methods are based on the *supervised autoencoder* (SAE) method [1].

We benchmark ATA and our SAE variants against MLPs (imbalanced classification baseline) and OCAs (outlier detection baseline) on various datasets, isolating aspects of imbalanced classification, outlier detection, and novelty detection. We find that the integration of contextual outliers in ATA significantly improves the outlier detection performance in comparison to its semi-supervised counterpart OCA. Furthermore, in contrast to the baselines, we show that ATA consistently achieves competitive performance on imbalanced classification, contextual outlier detection and novelty detection. SAE’s performance is comparable to ATA’s results.

This chapter is based on our publications [23, 24] and provides a solution to the first milestone, introduced in Sec. 1.5. The initial idea in [23] of extending OCA towards ATA by enriching the

outlierness signal of the reconstruction error via reconstruction error maximization of outlier samples was contributed and implemented by Max Lübbering during a customer project at Fraunhofer IAIS. The design of the experimental setup was discussed among all co-authors and implemented by Max Lübbering. The publication was written by Max Lübbering in most parts, who was supported by the co-authors discussing the paper idea, revising the paper, and researching the related work.

Similarly, in our publication [24], the initial idea of introducing MTL as a supervised classification task for improving the learned hidden representation has been contributed and implemented by Max Lübbering. The training, evaluation, and benchmarking of the approach were thoroughly discussed among all peers. Max Lübbering conducted all experiments. The resulting paper was written by Max Lübbering, who was supported in its revising by the co-authors, who also contributed to the related work and introduction.

In both publications all the authors of both publications participated in the frequent discussion of the results, and critically challenged the modeling, training, and evaluation approach.

3.1 Introduction

Standard DNN methods mostly assume that the number of training examples pertaining to different categories are roughly equal. However, in real-world scenarios, the distribution of categories is often rather skewed, with some group of data (majority class samples) occurring more frequently than others (minority class samples), leading to the imbalanced dataset problem [127]. These problems are common in medical diagnosis [128], fraud detection [129, 130], image classification [131, 132], etc.

The imbalanced data problem poses several challenges. Firstly, it has significant effects on the performance of classifiers [128, 133]. Secondly, the skewed distribution in data induces a bias in learning algorithms, pushing them towards predicting only the majority group, which is a known problem of fairness and bias [134]. From a data mining perspective, the minority class is more crucial in many applications, e.g., detection of seizures, arrhythmia. Due to the limited availability of training samples for these events, classifiers may fail to detect these rare patterns.

A plethora of approaches for handling imbalanced datasets have been proposed, which fall under two main categories. Firstly, data-level methods focus on balancing the skewed data distribution either by under-sampling of the majority samples [135] or, more commonly, by over-sampling of minority samples [60, 135]. Secondly, algorithm-level methods focus on modifying the learning algorithm directly to facilitate the learning of minority samples. These methods adjust the decision threshold [136], and assign different costs [137] to the minority samples. Additionally, anomaly detection methods [138–140], e.g., based on autoencoders or support vector machines, can learn a representation of the majority class. Deviations from the majority representation, e.g., in terms of reconstruction error, can be leveraged to distinguish the minority and majority samples.

Methods for handling imbalance are closely related to outlier detection problems when looking at the one-class solution or thresholding [141–143]. The relatedness stems from the similarity in characteristics for both machine learning problems. Firstly, outliers are generally a minority in a dataset. Secondly, they are generated by a different underlying mechanism than the majority class, which also applies for a minority class in a supervised setting [42, 144, 145]. Furthermore, outlier detection is generally seen as an unsupervised problem, which distinguishes it from the imbalanced data problem the most. It becomes a supervised problem if data is labeled as “normal” and “abnormal” in the training set, to detect outliers in the test data. This is referred to as supervised outlier detection

or classification based anomaly detection, which one of its subproblems is dataset imbalance [43, 146].

3.2 Autoencoders for Outlier and Novelty Detection

Generally, a DNN-based classifier with weights Θ can be represented by the function $f : \mathbf{x} \mapsto \hat{y}$, mapping sample $\mathbf{x} \in \mathbb{R}^m$ to prediction \hat{y} , where $y, \hat{y} \in [0, 1]$ in case of binary classification and $y, \hat{y} \in [0, 1]^{|C|}$ with $\sum_{i=1}^{|C|} y_i = 1$ and $\sum_{i=1}^{|C|} \hat{y}_i = 1$ in case of multi-class classification with $|C|$ classes.

An autoencoder (AE) is a special type of neural network that aims to accurately reconstruct a given input vector, despite its bottleneck (i.e., hidden middle layer with reduced number of neurons) limiting information flow. Because of the bottleneck, the AE can only focus on the most informative features to achieve accurate reconstructions, and needs to disregard uninformative/correlated features. This setup forces the AE to map the data onto a low-dimensional manifold within its latent space, as explained in Sec. 1.3. Architecturally, the encoder part $e : \mathbb{R}^m \mapsto \mathbb{R}^k$ maps a given input \mathbf{x} to a lower dimensional representation \mathbf{h} in latent space (i.e., $m \gg k$). A decoder d reconstructs sample \mathbf{x} from \mathbf{h} by mapping \mathbf{h} back to the original space $g : \mathbb{R}^k \mapsto \mathbb{R}^m$. Thus, the entire AE architecture is defined as

$$\mathbf{h} = e(\mathbf{x}; \Theta_e) \quad (3.1)$$

$$\hat{\mathbf{x}} = d(\mathbf{h}; \Theta_d), \quad (3.2)$$

where Θ_e and Θ_d denote the network weights of the encoder and decoder, respectively.

An AE is optimized for accurate reconstruction using a reconstruction error criterion such as the mean squared error

$$L_{\text{MSE}}(\mathbf{x}, f(\mathbf{x}; \Theta)) = \frac{1}{m} \sum_{i=1}^m (x_i - f(\mathbf{x}; \Theta)_i)^2. \quad (3.3)$$

The AE-based outlier detector leverages the mean squared error as an outlierness signal. Intuitively, the reconstruction error of the majority class m^+ is optimized more heavily than the minority class m^- , which is underrepresented in the unsupervised case, and unconsidered in the semi-supervised case of the *one-class autoencoder* (OCA) [64, 147, 148]. As a result, the reconstruction error tends to be higher for minority samples compared to majority samples, making the reconstruction error a highly predictive signal of the outlierness of a sample. In practice, a threshold t on the reconstruction error is determined through a brute force line search, e.g., w.r.t. F1 score, to detect minorities with a satisfactory performance [144, 149].

$$y(\mathbf{x}, \hat{\mathbf{x}}) = \begin{cases} m^-, & L_R(\mathbf{x}, \hat{\mathbf{x}}) \geq t \\ m^+, & \text{otherwise.} \end{cases} \quad (3.4)$$

As we will show in the following sections, the reconstruction error L_R is an insufficient outlierness signal when majorities and minorities correlate in the feature space. As a solution, we will introduce a supervised outlier detection algorithm based on AEs that actively maximizes the reconstruction error of minorities at training time, thereby guiding the network to learn an application-specific outlierness signal based on the reconstruction error.

3.3 Related Work

Over the years, various deep learning methods have been proposed for outlier detection, and recent surveys of such methods can be found in [45, 46]. Early autoencoder-like networks for outlier detection have been introduced by Hawkins et. al. as replicator networks [64]. These networks have been used further in a one-class fashion [147, 148]. Autoencoders have been shown to generalize better than principal component analysis (PCA), reconstruct non-linear relations easily, and perform better in higher dimensions than support vector machines. In the past years multiple approaches for outlier detection involving autoencoders have been introduced [44, 141, 150]. The most prominent ones are robust deep autoencoders, which isolate outliers during training by using a modified loss function [65].

For the subproblem of imbalanced classification within deep learning-based outlier detection, two approaches are commonly used to address it, namely, data-level methods and algorithmic methods [59]. Data-level methods usually comprise class-specific over- and undersampling [60, 151], while algorithmic-level methods typically focus on cost-sensitive learning [152, 153] or novel loss functions [154]. Additionally, autoencoders have been recently used to modify input features to improve classification performance and combat class imbalance [155].

3.4 Evaluation

For imbalanced classification, we selected three prominent benchmark datasets, namely the Reuters dataset¹, the Arrhythmia dataset² and the ATIS dataset³. Similarly, for outlier detection and novelty detection, the KDD dataset⁴ is a commonly adopted dataset. Following other researchers' work, we derived four sub-datasets from KDD, resulting in a total of 7 benchmark datasets. Since our algorithms deal with binary classification tasks, each dataset is binarized into the classes *minority* and *majority* as shown in Table 3.1. As part of the preprocessing, the textual samples are vectorized using the well-established Glove word embeddings [156]. The Glove language model maps tokens into a 100-dimensional space based on word co-occurrences. The low dimensional representation space allows for the training of shallow neural networks on less data, without overfitting. In this and the following chapters, we often deal with small-scale textual datasets, and therefore prefer Glove embeddings over large language models with their high-dimensional representation spaces. For our work, the Glove word embeddings are pooled to obtain document representation, thus providing a 100-dimensional dense vector. Additionally, categorical features are converted to one-hot encoded vectors, and real-valued features are z-transformed.

Next, we detail the various datasets used in this chapter.

Reuters dataset This NLP dataset contains financial documents published by the Reuters newswire in 1987. A single document is assigned to at least one of the pre-defined 90 classes. Combined with the high class imbalance in this dataset, this is a standard benchmarking dataset for multi-label document classification as well as outlier detection [157–159]. However, since multi-label classification is out of

¹<http://www.daviddlewis.com/resources/testcollections/reuters21578/>

²<http://archive.ics.uci.edu/ml/datasets/Arrhythmia>

³<http://www.ai.sri.com/natural-language/projects/arpa-sls/atis.html>

⁴<https://www.unb.ca/cic/datasets/nsl.html>

Dataset	Type	Majority	Minority	Subclass share	Train		Val		Test	
					# m^+	# m^-	# m^+	# m^-	# m^+	# m^-
KDD	Outlier imbal.	NORMAL	U2R, R2L	all	47122	745	20221	302	9711	2236
	Outlier bal.	NORMAL	U2R, R2L, DOS, PROBE	all	47122	41059	20221	17571	9711	9083
	Novelty imbal.	NORMAL	U2R, R2L	none	47122	745	20221	302	9711	716
	Novelty bal.	NORMAL	U2R, R2L, DOS, PROBE	none	47122	41059	20221	17571	9711	3750
Reuters	Imbal	EARN, ACQ, TRADE, INTEREST MONEY-FX, MONEY-SUPPLY	RESERVES	all	3613	25	1609	12	2051	12
ATIS	Imbal	FLIGHT	QUANT, AIRFARE, ABBR GSERVICE, REST, APORT ALINE, CITY, F_NO, F_TIME G_FARE, F_AIRFARE DIST, AIRCRAFT, CAPA	all	3173	1101	423	149	424	162
ARR	Imbal	NORMAL	OTHERS (15 classes)	all	122	31	62	15	61	15

Table 3.1: Datasets and their subdatasets: Majority and minority frequencies for train, validation and test split. A subclass share of all means that *all* subclasses are shared between the dataset splits, whereas a subclass share of *none*, indicates that none are shared between train/val and test split. This is an important requirement for novelty detection. Note that m^+ and m^- refer to the majorities and minorities, respectively.

scope of this thesis, all documents having multiple labels were filtered out. Therefore, this dataset is representative for imbalanced classification.

ATIS dataset This dataset contains transcribed queries that passengers requested to the air travel information system (ATIS) to receive flight related information. The resulting *ATIS Spoken Language Systems Pilot Corpus* was labeled with a total of 17 classes. The prevalent class frequency imbalance makes this dataset a suitable candidate for imbalanced classification problems.

ARR dataset This dataset contains samples on heart arrhythmia data categorized into 18 classes. Due to its imbalanced nature and small size, this dataset is generally considered a difficult dataset for imbalanced classification.

KDD dataset This dataset consists of benign network communication samples, as well as four types of network intrusions. Due to the low prevalence of those intrusions and their diversity, this is a common dataset to benchmark outlier and novelty detection problems [141, 160–162]. Notably, the original KDD dataset had various inherent issues, including redundant features and information leakage. For this reason, we consider an improved version [163] that is devoid of these issues. The

intrusion types contain several subtypes and these subtypes are either shared by all splits or are only present in either the train/validation sets or the test set. Therefore, four datasets are derived with a combination of outlier/novelty and balance/imbalance, as shown in Table 3.1.

UMAP Visualization To further support the different nature of these datasets, we projected the samples' features onto a two dimensional plane using UMAP, an unsupervised manifold learning based algorithm for nonlinear dimensionality reduction [164]. Fig. 3.2 shows the results for the datasets Reuters, ATIS, and ARR for the imbalanced classification problem. The Reuters dataset expresses well separated clusters, whereas the minority and majority of the clusters majorly overlap on the ATIS dataset. For the ARR dataset, a single coherent cluster is formed with no separation between minority and majority samples, making this dataset the most difficult among the three imbalanced datasets. Similarly, the clusters of the four KDD based outlier and novelty datasets are displayed in Fig. 3.1. Since, as per definition, novelties only appear in the test set, the UMAP model was trained on the training split first and subsequently applied to the train and test split separately. Here, for the balanced case, two insights can be deduced: a) the outliers and inliers form proper clusters on both splits, and b) novelties do not cluster. Both insights support the definitions of outliers and novelties, thus making these datasets valid representatives for the respective classification problem.

3.5 From Imbalanced Classification to Supervised Outlier Detection Problems

After the previous introductory sections in this chapter, we are concerned with supervised outlier detection and its subproblem of imbalanced classification in this section. In particular, we devise solutions that can be applied to both supervised outlier detection and imbalanced dataset problems. In particular, we focus on the autoencoder approach, using reconstruction error as an informative feature for classification. Typically, only majority samples are used for fitting the autoencoder, which is referred as *one-class autoencoders* (OCA). However, this approach becomes ineffective when the majority and minority samples overlap in the feature space of the AE, or minority samples are learned in later epochs by the AE, due to minority contamination in the training set [165]. In this case, the corresponding loss distributions also overlap, and therefore prevent us from accurate discrimination. To address this limitation, we propose an adversarial style of training autoencoders. The main idea is, instead of training the autoencoders to minimize only the reconstruction loss for majority samples, they can also be trained to maximize the loss for minorities, thereby enriching the reconstruction error signal for classification.

Our main contributions are as follows: (i) We introduce *adversarially trained autoencoders* (ATA) for imbalanced classification, as well as supervised outlier and novelty detection problems. (ii) We empirically show that ATA outperforms two baselines, namely OCA and a multilayer perceptron (MLP), in terms of AUROC and F1 score. (iii) While the baselines show task dependent performances, ATA provides high robustness across all three tasks.

3.5.1 Adversarially Trained Autoencoders

We introduce adversarially trained autoencoders (ATA), a novel approach that builds upon OCA's idea of leveraging the reconstruction loss as an expressive feature for classification. While architecturally

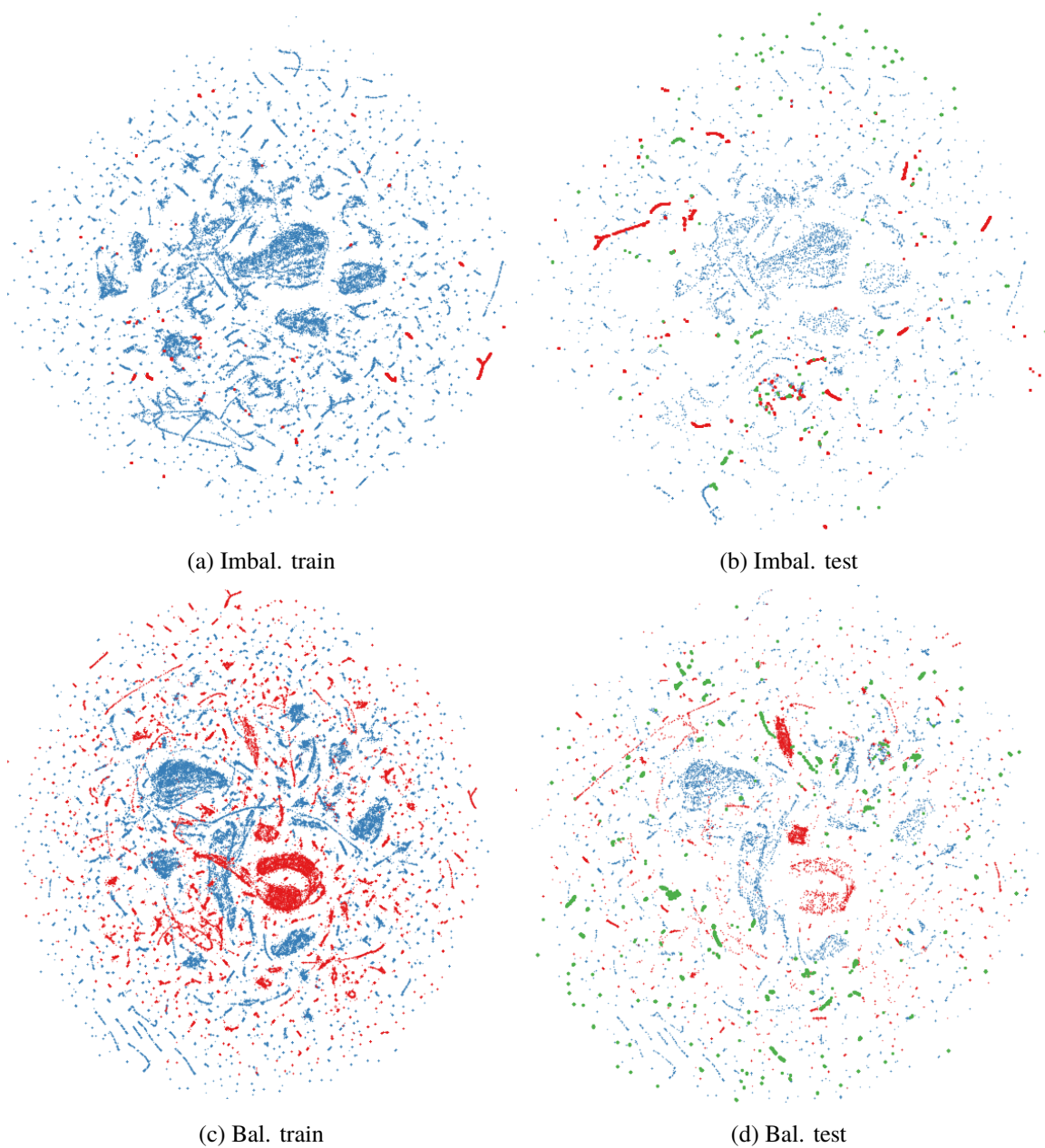


Figure 3.1: **KDD outlier and novelty datasets:** Visualization of the train and test splits after reducing the dimensionality to 2D using UMAP [164]. Outlier samples are highlighted in red, inliers in blue and novelties in green. Fig. 3.1(a) and Fig. 3.1(b) show the clustering of the imbalanced variant of datasets, while Fig. 3.1(c) and Fig. 3.1(d) show the clustering for the balanced datasets. Note that, as per definition, novelties only appear in the test set, whereas outliers appear in both.

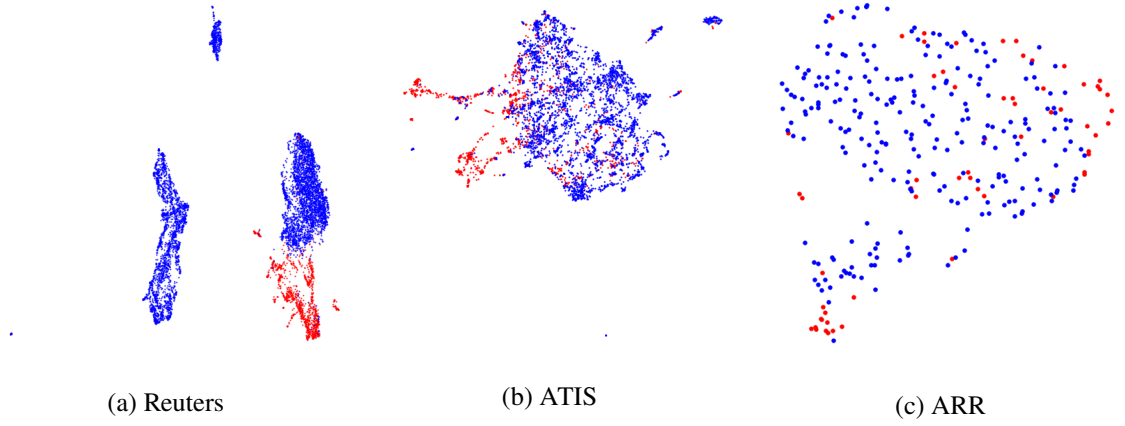


Figure 3.2: Imbalanced datasets: Visualization of the Reuters, ATIS, and arrhythmia datasets after projecting the samples onto a two-dimensional plane, using UMAP [164] for dimensionality reduction. Minorities samples are highlighted in red, majorities in blue.

similar, we propose a new training style that additionally incorporates minority samples, making this a supervised classifier. ATA intends to resolve OCA’s aforementioned deficiency of working poorly in imbalanced classification scenarios, and MLP’s deficiency concerning novelty and outlier detection tasks, by introducing an adversarial loss function specifically engineered to deliver robust results in each of these domains.

This loss function not only minimizes the reconstruction loss for majority samples, but also maximizes the loss for minority samples as defined by

$$L_{adv}(\mathbf{x}, \hat{\mathbf{x}}, y) = \begin{cases} L_R(\mathbf{x}, \hat{\mathbf{x}}) \cdot 0, & L_R \in [l, u] \quad \wedge \quad y \in \text{minority} \\ L_R(\mathbf{x}, \hat{\mathbf{x}}), & L_R(\mathbf{x}, \hat{\mathbf{x}}) > u \quad \vee \quad y \in \text{majority} \\ -\alpha L_R(\mathbf{x}, \hat{\mathbf{x}}), & \text{otherwise.} \end{cases} \quad (3.5)$$

While the loss for majority samples, i.e., target $y \in \text{majority}$, is calculated as the plain reconstruction loss, the reconstruction loss for minority samples is adapted depending on its magnitude. Our objective is to maximize the minority reconstruction loss until it falls within the pre-defined range $[l \in \mathbb{R}, u \in \mathbb{R}]$, thus effectively clipping the maximum loss of minority samples. If a minority sample’s loss already falls within this range, then we clear out its gradient by multiplying the loss with 0 (first case). If the loss value is above the range, we minimize it, as we do for the majority samples (second case). Minimization of losses $\gg u$ prevents exploding gradients. If the loss is below range $[l, u]$, the loss is multiplied by the negation of the minority weighting factor $\alpha \in \mathbb{R}^+$. While not immediately apparent, negation of the loss function corresponds to flipping the gradient as stated by Theorem 1. With this adversarial approach, it becomes easy to find a threshold afterwards that discriminates the samples efficiently. We refer to the autoencoder trained in this fashion as an *adversarially trained autoencoder* (ATA).

Theorem 1. *The gradient $\nabla_{\Theta} L_{adv}$ for minority samples acts in opposing directions to keep the loss of minority samples in the defined bin $[l, u]$.*

Proof. Let $\mathcal{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}$ be a set of n samples, where each sample $\mathbf{x}^{(i)} \in \mathbb{R}^d$. The corresponding

targets are denoted by the set $Y = \{y^{(1)}, \dots, y^{(n)}\}$, where each target $y^{(i)} \in \{m^+, m^-\}$ with minorities being denoted by m^- and majorities by m^+ . The autoencoder network $f : \mathbf{x} \mapsto f(\mathbf{x}; \Theta)$ parameterized by weights Θ reconstructs a sample \mathbf{x} .

Then, the gradient of the overall loss L_{total} computes to

$$\nabla_{\Theta} L_{\text{total}}(f, \mathcal{X}, Y) = \nabla_{\Theta} \sum_{i=1}^n L_{\text{adv}}(f_{\Theta}(\mathbf{x}^{(i)}), y^{(i)}) \quad (3.6)$$

$$= \nabla_{\Theta} \left(\underbrace{\sum_{\{i|y^{(i)} \in m^+\}} L_R(f(\mathbf{x}^{(i)}; \Theta), y^{(i)})}_{\text{majority rec. loss}} + \underbrace{\sum_{\{i|y^{(i)} \in m^-\}} L_{\text{adv}}(f(\mathbf{x}^{(i)}; \Theta), y^{(i)})}_{\text{minority rec. loss } L_{m^-}} \right). \quad (3.7)$$

The gradient for the minority reconstruction loss, which is the second addend, can thus be expressed by

$$s_1 = \{i | y^{(i)} \in m^- \wedge L_R(f(\mathbf{x}^{(i)}; \Theta), y^{(i)}) \in [l, u]\} \quad (3.8)$$

$$s_2 = \{i | y^{(i)} \in m^- \wedge L_R(f(\mathbf{x}^{(i)}; \Theta), y^{(i)}) < l\} \quad (3.9)$$

$$s_3 = \{i | y^{(i)} \in m^- \wedge L_R(f(\mathbf{x}^{(i)}; \Theta), y^{(i)}) > u\} \quad (3.10)$$

$$\begin{aligned} \nabla_{\Theta} L_{\text{adv}}(\Theta, \mathbf{x}, y) &= \nabla_{\Theta} \left(\sum_{s_3} L_R(f(\mathbf{x}^{(i)}; \Theta), y^{(i)}) \right) \\ &\quad + 0 \cdot \nabla_{\Theta} \left(\sum_{s_1} L_R(f(\mathbf{x}^{(i)}; \Theta), y^{(i)}) \right) \\ &\quad - \alpha \nabla_{\Theta} \left(\sum_{s_2} L_R(f(\mathbf{x}^{(i)}; \Theta), y^{(i)}) \right) \end{aligned} \quad (3.11)$$

□

The complete training algorithm for ATA is presented in Alg. 1, and the procedure L_{adv} is a direct implementation of Eq. 3.5. The training routine for a single epoch is implemented in the procedure *TRAIN_EPOCH*, and the dataset for training is split into batches. While iterating over the batches, a batch's samples X are reconstructed by the autoencoder f_{Θ} as denoted by reconstructions \tilde{X} . Then, the respective losses L_{adv} are computed from the samples and reconstructions. Finally, the gradient of L_{adv} is calculated w.r.t. the network parameters Θ by backpropagation and parameters Θ are updated in the gradient step. Once the network is trained, the classification threshold β is found through a brute-force line search by trying out different threshold values and selecting the one which has the highest F1 score on the validation set.

In summary, the intuition behind our adversarial loss function is to minimize the loss L_R of majority samples, as done by OCA for outlier and novelty detection. However, this approach immediately fails when the loss of minority samples is mistakenly minimized as well. This is a common issue, for instance, when minority and majority samples are highly correlated in feature space, but the loss maximization of L_{adv} successfully addresses this issue by enforcing loss maximization of minority samples. Of course, there is a limit to it, when samples are virtually indistinguishable, causing any machine learning approach inherently to fail.

Algorithm 1 Adversarial Training of ATA**Input:** sample \mathbf{x} , reconstruction $\hat{\mathbf{x}}$, target y , minority weighting factor α , minority loss range $[l, u]$

```

procedure  $L_{\text{ADV}}(\mathbf{x}, \hat{\mathbf{x}}, y, \alpha, l, u)$ 
   $L_R \leftarrow \frac{1}{n} \sum_{i=1}^n (x_i - x'_i)^2$ 
  if  $y$  is minority then
     $L \leftarrow \begin{cases} L_R \cdot 0, & L_R \in [l, u] \\ L_R, & L_R > u \\ -\alpha L_R, & \text{otherwise} \end{cases}$ 
  end if
  return  $L$ 
end procedure

```

Input: Sample set χ , Target set Y , Batch size b , Autoencoder f with parameters Θ , lr λ

```

procedure TRAIN_EPOCH( $\chi, Y, b, f, \lambda$ )
  Generate batches from  $\chi$  each of size  $b$ 
  for each  $X', Y' \in$  batches do ▷ for each batch
     $\hat{X}' \leftarrow \{f(\mathbf{x}^{(i)}; \Theta) \mid \forall \mathbf{x}^{(i)} \in X'\}$  ▷ forward pass
     $L \leftarrow \{\}$  ▷ initialize batch loss
    for  $i \leftarrow 1, \dots, b$  do
       $L \leftarrow L \cup \{L_{\text{adv}}(\mathbf{x}^{(i)}, \hat{\mathbf{x}}^{(i)}, y'^{(i)})\}$  ▷ sample loss
    end for
    Calc. gradient  $\mathbf{g} \leftarrow \nabla_{\Theta}(\sum_{i=0}^b L_i)$  ▷ backprop
     $\Theta \leftarrow \Theta - \lambda \mathbf{g}$  ▷ gradient update step
  end for
end procedure

```

3.5.2 Experiments and Results

In this section, we discuss the experiments and performance of our approaches on the different datasets introduced in Sec. 3.4, which have been selected to individually represent one of the previously described aspects of imbalanced classification, supervised outlier and novelty detection. Here, the models are evaluated with respect to AUROC and F1 score. As explained in Sec. 1.2.1 the AUROC metric corresponds to evaluating a model at all possible thresholds, resulting in a score that is independent of thresholds. Thus, this metric is highly compelling for outlier and novelty detection, as a concrete threshold is generally chosen based on the use case. To evaluate the models w.r.t. imbalanced classification problems, macro F1 score is used for evaluation.

To benchmark our ATA approach, we consider the baselines of MLP (including oversampling) and OCA as they are standard approaches in tackling imbalanced and outlier detection tasks, respectively. For a fair comparison, we keep the model complexity roughly similar. The MLP has a single hidden layer of size 50 also with sigmoid activations and an output layer with one neuron. The autoencoder of OCA and ATA models has a single hidden layer of size 50, also with sigmoid activations. Note, the weights of AE are not tied as we did not witness any significant performance differences. Furthermore, we applied a weight decay of $1e^{-4}$ to all models to prevent overfitting and the exploding of gradients.

3.5 From Imbalanced Classification to Supervised Outlier Detection Problems

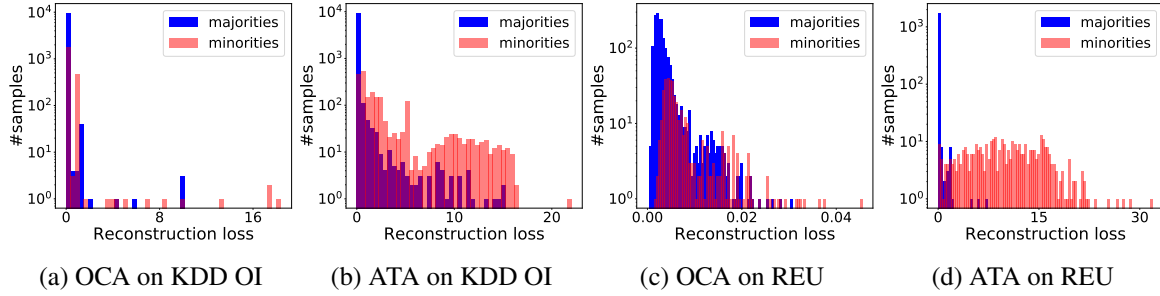


Figure 3.3: Loss histograms of the best OCA models and ATA models on the Reuters and KDD OI dataset. The explicit reconstruction error maximization of minority samples leads to better separation of majorities and minorities in reconstruction error space. Interestingly, OCA leads to better reconstruction of majorities, but implicitly also minimizes minorities, resulting in poor outlier detection performance.

Models	Imbalanced			Outlier			Novelty							
	ATIS	REU	ARR	KDD OI	KDD OB	KDD NI	KDD NB							
	AUROC	F1 AUROC	F1 AUROC	F1 AUROC	F1 AUROC	F1 AUROC	F1 AUROC	F1	F1					
ATA	99.07%	95.77%	99.80%	97.76%	92.26%	88.17%	97.36%	86.73%	97.10%	89.26%	82.34%	57.18%	92.11%	70.86%
MLP	99.00%	94.64%	99.82%	97.34%	89.46%	89.38%	93.93%	78.22%	93.96%	89.06%	61.54%	55.68%	89.31%	73.45%
OCA	86.17%	74.51%	85.42%	71.26%	80.65%	75.30%	92.75%	83.88%	95.68%	90.60%	80.48%	70.83%	93.15%	84.93%

Table 3.2: AUROC and macro F1 scores of the best ATA, MLP, and OCA models on the seven datasets. It is observed that ATA outperforms baseline methods in most of the tasks, depicting its robustness.

For the MLP and ATA, we allowed for balanced sampling of each class as a counter measure to class imbalance (see Sec. 1.2.3), which by design is not applicable to OCA. The models are trained using the Adadelata optimizer [31] which is less prone to saddle points than SGD [30] (see Sec. 1.1). We performed a formal grid search of hyperparameters concerning learning rate, balanced sampling, and outlier weighting factor α and bin range $[l, u]$. Once finished, we selected the best model for MLP, OCA, and ATA based on the AUROC score on the validation set.

Table 3.2 summarizes the performance of selected best models on each dataset, and by comparing the AUROC scores, we observe that ATA outperforms each baseline in 5 out of 7 tasks. Another important observation is that MLP and OCA excel in different types of tasks. While MLP is strong on imbalanced classification tasks, its performance degrades on outlier and novelty detection tasks. Au contraire, OCA excels on outlier/novelty detection but performs much worse on imbalanced classification tasks. This task dependent performance degradation is not prevalent for ATA, and, in fact, not only does ATA outperform MLP on imbalanced classification and perform equally with OCA on novelty tasks, it also clearly outperforms both baselines on the intermediate outlier detection task.

Our results clearly indicate that the ATA approach is robust in all settings of imbalanced classification and supervised outlier detection, even in semi-supervised novelty detection tasks. This is due to the adversarial autoencoder training for minimizing and maximizing the reconstruction losses. As captured in Fig. 3.3, ATA provides a clear separation of majority and minority loss distributions

irrespective of the task at hand. This property of ATA makes it overall a highly compelling tool for supervised classification and outlier detection tasks.

3.5.3 Conclusion

With the *adversarially trained autoencoder* (ATA), we proposed a new architecture for outlier and novelty detection. Jointly maximizing/ minimizing the reconstruction loss for majorities/minorities, respectively, resulted in highly robust models excelling on imbalanced classification, supervised outlier detection, and semi-supervised novelty detection. This level of robustness is unparalleled by any of the baselines, all of which fail on at least one of the tasks. This opens up several promising directions for future work. Firstly, we could replace the brute-force routine of thresholding by a classification layer that takes reconstruction loss as input. In this way, we could train the whole network in an end-to-end fashion, with multiple objectives by extending ATA with supervised autoencoders [1]. We follow the direction of supervised autoencoders in the following section. Secondly, to extend our ATA approach to unsupervised outlier detection problems, we could split our training into two phases. In the first phase, autoencoders could be trained classically, to minimize reconstruction loss. The second phase would be a fine-tuning phase, in which we also maximize the reconstruction loss for potential anomalies, which are samples with higher reconstruction errors.

3.6 Supervised Autoencoder Variants for End-to-end Anomaly Detection

Learning from limited and imbalanced training data and generalizing to unseen data and corruptions, is an actively studied research area in deep learning [166, 167]. In addition to common strategies to avoid overfitting, such as weight decay, drop out, and early stopping, an alternative direction has emerged in the form of multitask learning (MTL) [168]. The idea is to incorporate additional tasks into the learning process to improve the generalization of the model and to learn from a reduced number of samples. These additional tasks mostly focus on including unsupervised objectives [169, 170] to learn better representations within the intermediate layers of a deep network.

Especially in supervised outlier detection, the low base rate of outliers and minority samples easily lead to overfitting and poor quality representations in embedding space. If these representations were used for downstream outlier classification, the performance of the outlier detectors would be severely impaired. To this end, we explore MTL to learn richer representations by forcing the model to solve a related, self-supervised problem of high complexity that retains the problem complexity in embedding space. Analogously to ATA, we force the network via MTL to map the high dimensional inlier data generating process onto a low dimensional manifold that approximates the true latent data manifold of the inlier data generating process. In this section, we explore this direction using *supervised autoencoders* (SAEs) [1].

We pursue the direction of MTL by training a SAE to jointly classify and reconstruct a sample. Typically, in this setting, while minimizing the auxiliary reconstruction loss, either all the samples (vanilla AEs) or just the majority samples (one-class AEs) are used during training. Both approaches become ineffective when majority and minority samples overlap in the feature space and prevents the classifier from accurate discrimination. To address this limitation, we propose an auxiliary task based on the adversarial style of training autoencoders introduced in Sec. 3.5. The main idea is instead

of training the autoencoders to minimize only the reconstruction loss for majority samples, they can also be trained to maximize the loss for minorities, thereby enriching the embedded features for classification.

Therefore, our main contributions are:

1. We introduce three novel, autoencoder-based end-to-end approaches by adaption of the vanilla SAE architecture. Namely, a) *adversarially trained supervised autoencoders* (ASAE), which incorporates an adversarial loss function instead of the mean squared error loss, b) *supervised autoencoders with reconstruction loss* (SAER) which forwards the reconstruction loss to the classification layers as a predictive feature and c) *adversarial supervised autoencoders with reconstruction* (ASAER) which is a combination of ASAE and SAER.
2. The proper functioning of all SAE variants is thoroughly verified by empirically analyzing the reconstruction loss distribution and clustering of the autoencoders' encoding.
3. Additionally, we show the superiority of our methods compared to potent autoencoder methods and a vanilla MLP on the seven datasets for the imbalanced classification, novelty and anomaly detection problems introduced in Sec. 3.4.

3.6.1 Supervised Autoencoders

Supervised autoencoders (SAEs) introduced by Le et al. [1] improve the generalization performance of neural networks by adding an unsupervised auxiliary task to the supervised learning task. From an architectural point of view, the neural network model is a combination of an AE performing the auxiliary task by reconstructing the inputs, and a neural network performing the classification task, as shown in Fig. 3.4. Notably, both networks share the encoding layers. The overall architecture is trained in an end-to-end fashion by minimizing a combined loss function L_{total} , that is composed of both the reconstruction loss L_R taking sample \mathbf{x} and reconstruction $\hat{\mathbf{x}}$ as its inputs and the classification loss L_C taking target y and prediction \hat{y} as its inputs:

$$L_{\text{total}}(\mathbf{x}, \hat{\mathbf{x}}, y, \hat{y}) = \lambda L_C(y, \hat{y}) + L_R(\mathbf{x}, \hat{\mathbf{x}}). \quad (3.12)$$

Notably, this loss function does not introduce any scaling of L_C and L_R , thus making the effect of the weighting hyperparameter λ dependent on the scale of the inputs. While the authors state that $\lambda = 0.01$ provided the best results, it is not interpretable whether this factor equalizes both losses or favors one of them.

3.6.2 Adversarial Supervised Autoencoders

Building upon SAEs, we introduce three novel variants specifically targeted at tackling imbalanced dataset, outlier detection and novelty detection problems. While the authors of SAE show that their architecture is less prone to overfitting and yields better generalization performance on the classification tasks, we investigate whether SAEs and our proposed variants can achieve robust results on imbalanced classification, outlier detection and novelty detection.

Our work builds upon the idea of exploiting the reconstruction loss as a highly predictive feature for anomaly detection, which we already investigated in Sec. 3.5 and [23] as part of our *adversarially*

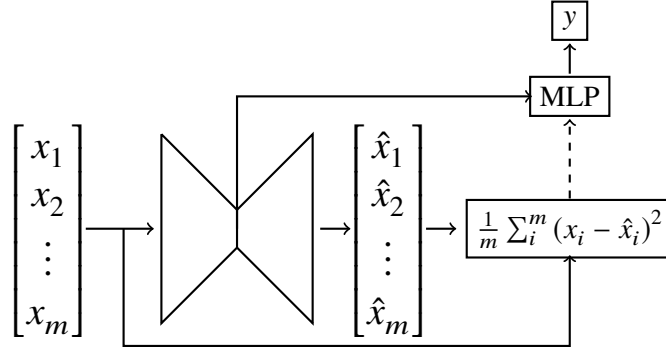


Figure 3.4: Conceptual design of a *supervised autoencoder* (SAE) and a supervised autoencoder with readout loss (SAER). In both variants the autoencoder maps the input sample \mathbf{x} to the reconstruction $\hat{\mathbf{x}}$. In the SAE case, the MLP performs inference solely on the encoding of \mathbf{x} , whereas in the case of SAER, the reconstruction loss is also passed to the MLP, as indicated by the dashed line.

Model	L_R	L_R to MLP
SAE	L_{MSE}	no
SAER	L_{MSE}	yes
ASAE	L_{R_adv}	no
ASAER	L_{R_adv}	yes

Table 3.3: Summarization of the different model variations. The models are distinguished on an architectural level based on whether the reconstruction loss is passed to the MLP or not, as defined in the third column. Additionally, there are two different training styles by adaptation of the reconstruction loss function L_R in Eq. (3.14): Mean squared error L_{MSE} and adversarial reconstruction loss L_{R_adv} .

trained autoencoder (ATA) architecture. As a recall, this architecture is trained in an adversarial supervised fashion, minimizing/maximizing the reconstruction loss for majorities and minorities, respectively, as formalized by:

$$L_{R_adv}(\mathbf{x}, \hat{\mathbf{x}}, y) = \begin{cases} 0, & L_{\text{MSE}} > t \wedge y \in m^- \\ L_{\text{MSE}}(\mathbf{x}, \hat{\mathbf{x}}), & y \in m^+ \\ -\alpha L_{\text{MSE}}(\mathbf{x}, \hat{\mathbf{x}}), & \text{otherwise.} \end{cases} \quad (3.13)$$

Given the loss function L_{R_adv} , ATA learns to reconstruct majority samples having class label m^+ , and the loss of minority samples is maximized up to a chosen threshold t . The maximization intensity is controlled by an outlier weighting factor $\alpha \in \mathbb{R}^+$. When the reconstruction loss exceeds this threshold t , the loss is zeroed out, thus zeroing out the gradients as well, resulting in no further learning w.r.t. the given sample. It has been shown in Sec. 3.5, that loss negation is equivalent to flipping the gradient.

By pushing the loss distribution of minority samples towards threshold t and the majority loss distribution towards 0, this adversarial training style enforces the predictive power of the reconstruction loss for anomalies. For estimating the decision boundary, we performed a brute force line search over

the reconstruction loss space to find a suitable decision threshold.

In this section, we combine the merits of reconstructive representation learning for outlier detection and MTL for expressive representation learning, by proposing two different extensions to SAE, whose combinations result in three novel methods, as shown in Table 3.3.

In contrast to the loss function of the original SAE, as defined in Eq. (3.12), we improve the loss functions in two ways. Firstly, we equalize the importance of each loss term L_c and L_R at the beginning of the training by unit scaling them with the scaling factors s_c and s_R , respectively. Secondly, we sum up a linear combination of both loss terms instead of penalizing a single term individually. With these two changes, as defined in Eq. (3.14), the influence of each loss term can be investigated empirically.

$$L_{\text{total}}(\mathbf{x}, \hat{\mathbf{x}}, y, \hat{y}) = \frac{\lambda}{s_c} L_c(y, \hat{y}) + \frac{1 - \lambda}{s_R} L_R(\mathbf{x}, \hat{\mathbf{x}}) \quad (3.14)$$

The first SAE extension replaces the reconstruction loss L_R within Eq. (3.14) with the adversarial reconstruction loss function L_{R_adv} , previously defined in Eq. (3.13). As shown in Table 3.3, models making use of the adversarial reconstruction loss function have a leading *A* in their name.

For our second SAE extension, the reconstruction loss L_R is passed to the MLP along with the encoding, as indicated by the dashed arrow in Fig. 3.4. As already shown for ATA and OCA, the reconstruction loss is a highly predictive outlierness feature, and therefore, is reasonable to be included in the input of the MLP. In Table 3.3, the presence of this property is indicated by a trailing *R* in the respective model names.

3.6.3 Experiments and Results

In this section, the experiment setup and the model performances on seven different datasets are discussed. We chose the same datasets and preparation from Sec. 3.5, representing all three tasks of imbalanced classification, supervised outlier, and novelty detection.

We consider the AUPR score as a threshold independent metric that, in contrast to AUROC, takes different base rates into account [171]. The metric can be interpreted as the proportion of samples to be predicted as minorities, over the set of all predictions exceeding a randomly selected threshold [172]. It is generally applied to problems dealing with "finding a needle in a haystack", and therefore, it is more relevant for outlier and imbalanced dataset problems.

To achieve a fair comparison, we keep the model complexity of each method roughly the same. Furthermore, the three baselines, such as ATA, MLP and OCA are chosen based on our previous work of ATA [23]. It is noteworthy that, due to the decoder in the ATA, OCA and SAE methods, these methods have more parameters when compared to an MLP. However, as shown in [23], tied weights between the encoder and decoder did not have any significant effect on the performance of these models. We use Adadelta [31] as an optimizer for our models, and additionally, to account for overfitting and exploding gradients, we regularize with a weight decay of $1e^{-4}$. With respect to the training objective, the MLP minimizes the samples' binary cross entropy, while OCA and ATA optimize the samples w.r.t. the mean squared error. SAE methods, on the other hand, optimize the multitask loss function as a linear combination of binary cross entropy for classification loss and mean squared error for the reconstruction loss. We perform a formal grid search for each model on each dataset w.r.t. learning rate and balanced sampling. Note that balanced sampling is not applicable to OCA since it is semi-supervised. Particularly for ATA, the grid search comprised different outlier

	Imbalanced			Outlier		Novelty	
	ATIS	REU	ARR	KDD OI	KDD OB	KDD NI	KDD NB
MLP	97.34%	99.10%	78.02%	77.27%	93.26%	14.48%	75.95%
OCA	75.09%	49.16%	50.91%	65.63%	92.51%	20.35%	76.37%
ATA	96.09%	99.32%	82.39%	74.86%	95.35%	20.66%	69.06%
SAE	98.51%	99.31%	80.97%	83.24%	94.73%	21.00%	61.44%
ASAE	98.10%	99.37%	79.53%	80.11%	93.50%	15.67%	56.23%
SAER	98.02%	99.29%	73.52%	80.53%	93.21%	19.78%	75.31%
ASAER	97.81%	99.43%	83.04%	77.87%	90.42%	21.34%	66.46%
BASE	27.65%	0.58%	19.74%	18.72%	48.33%	6.87%	27.86%

Table 3.4: Performance comparison of vanilla SAE, ASAE, SAER, and ASAER to the baselines MLP, OCA and ATA based on AUPR scores throughout the seven datasets (ATIS, REU, ARR, KDD OI, KDD OB, KDD NI and KDD NB). For each dataset, the score of the best model has been highlighted in bold face. It is observed that SAE approaches generally outperform the baselines. BASE denotes the expected AUPR scores of a random classifier for reference.

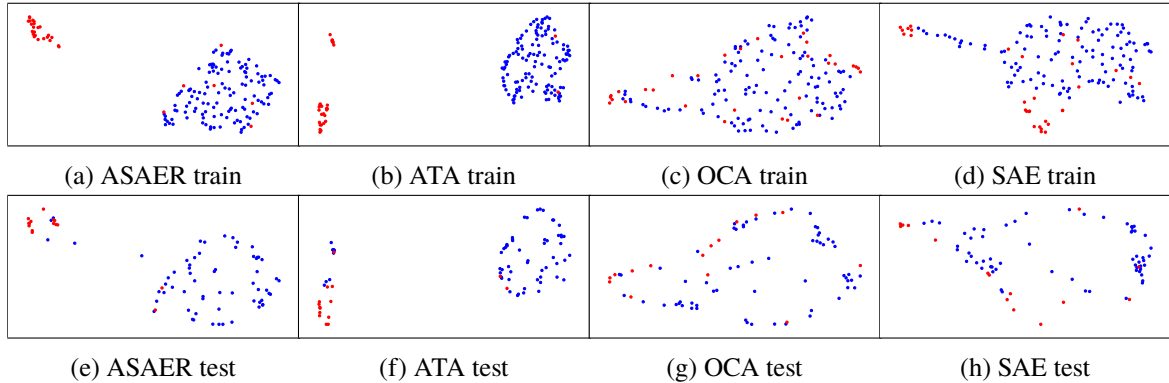


Figure 3.5: Scatter plots of the minority encodings (red points) and majority encoding (blue points) generated by the best ASAER, ATA, OCA and SAE models on the ARR dataset split by train and test. To visualize the encodings within \mathbb{R}^2 , we employed UMAP, an unsupervised manifold learning based algorithm for nonlinear dimensionality reduction [164]. In contrast to the non-adversarial methods, the results suggest that the adversarial methods successfully separate the minorities from the majority manifold. This becomes especially apparent when comparing the level of separation to the original clustering result on the raw samples in Fig. 3.2(c).

weighting factors α , loss term weighting factors λ , and minority reconstruction loss thresholds t . For every dataset and every method, we select the best model based on the highest AUPR validation score.

Table 3.4 summarizes the test performance of the selected best models on each dataset. Overall, it can be seen that the SAE method and its variants are superior to the baselines of MLP and OCA, as the SAE variants beat the MLP approaches in 21 out of 28 cases, and the OCA in 22 out of 28 cases in terms of AUPR score. Comparing the performance to ATA, the SAE variants beat ATA in 14 out of

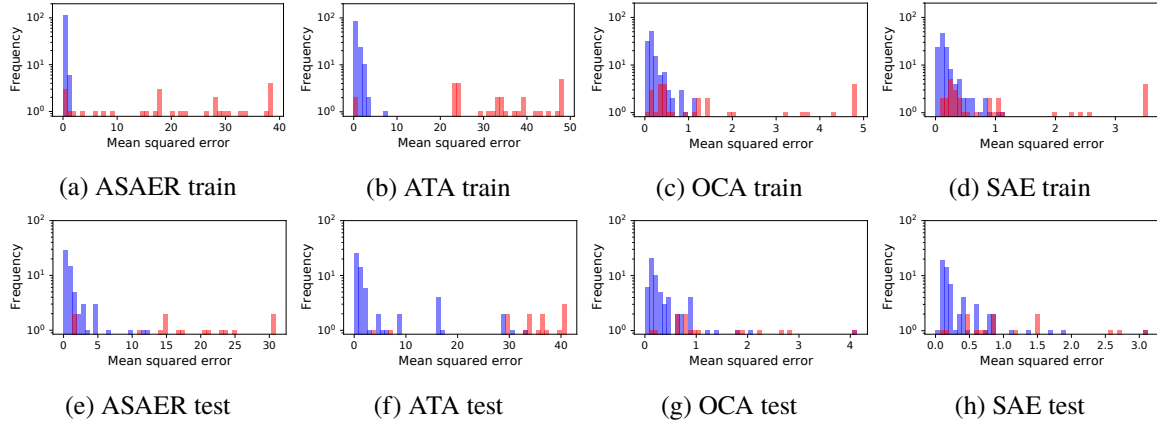


Figure 3.6: Histograms of reconstruction loss distributions of minority (red bars) and majority samples (blue bars) for the best ASAER, ATA, OCA and SAE models on the ARR dataset split by train and test. Note y-axis is log scaled.

28 cases, putting ATA at no architectural advantage. Furthermore, at least one of the SAE approaches exceeds all baseline approaches in 5 out of 7 datasets, with SAE’s and ASAER’s performance standing out. In conclusion, by direct AUPR score comparison, SAE and its variants are superior to MLP and OCA, and competitive to ATA throughout the imbalanced classification, outlier, and novelty detection problems.

Next, we explore the representations learned by these best models and visualize them using UMAP clusters, as seen in Fig. 3.5. We can observe that for the adversarial approaches, i.e., ASAER and ATA, the resulting clusters are well separated, with each cluster almost exclusively containing either inlier or outlier data points. However, for the non-adversarial approaches such as OCA and SAE, there is only a single cluster without any clear separation. This is a valuable insight, clearly showing that adversarial approaches force the autoencoder to learn better representations. Furthermore, we visualize the reconstruction loss distributions of outlier and inlier data points using a histogram as shown in Fig. 3.6. It can be seen that the loss distributions corresponding to inlier and outlier data points are strongly separated for adversarial methods. In contrast to this, the loss distribution overlaps for non-adversarial methods. This strong separation is due to the classification and reconstruction term within the adversarial loss functions of ASAER, as defined in Eq. (3.14); both objectives are aligned with the core objective of separating majorities from minorities, yielding the minorities being remote from the latent inlier manifold. In contrast, the SAE objective aims to minimize the reconstruction losses regardless of the sample’s class. In other words, this auxiliary task of unsupervised learning is not aligned to the classification task. Our work, therefore, suggests choosing an auxiliary task (e.g., adversarial loss minimization) that is similar to the primary task.

3.6.4 Conclusion

We developed three novel end-to-end variants of supervised autoencoders [1] which can be used for imbalanced classification, outlier detection, and novelty detection. Having evaluated our approaches on a broad spectrum of datasets against competitive baselines such as MLP, OCA, and ATA methods, we show that SAE and its variants are superior to MLP and OCA, and competitive to ATA, in

terms of AUPR scores. Additionally, the representations obtained from adversarial approaches show well-separated clusters suggesting the need for further investigation.

Our work indicates that auxiliary tasks relevant for classification task, help to obtain encodings useful for accurate classification. Furthermore, one could explore whether the reconstruction loss alone can be passed as a feature to the MLP without the encodings. This setup would automatically force the MLP to separate majority and minority samples based on reconstruction loss which may also render the adversarial objective redundant. This could happen because the classification loss might be sufficient. We would like to pursue this approach as future work; thereby, we can reduce the multi-task learning problem into a simple classification problem with this informed architecture. While this might seem straight forward, there are some limitations to this kind of optimization that lead to poor robustness when solely optimizing the classification loss in this setting, as we will demonstrate by the ablation study concerning the proposed loss function in Sec. 4.7.4.

3.7 Summary and Outlook

In this chapter, we explored deep learning-based supervised outlier detection from three angles. Firstly, we have demonstrated through various experiments that traditional DNNs are incapable of detecting outliers, novelties, and generally unobserved corruptions. If outlier data is available at training time, the DNN can leverage this information to treat the outlier detection problem as an imbalanced classification problem. However, this only works under the strong empirical risk minimization assumption, that demands the outliers observed at training to be representative of the ones observed at test time. We have shown that this is not always given, leading to severe performance degradation of traditional classification methods.

Secondly, we explored *one-class autoencoders* for semi-supervised outlier detection which yield strong outlier detection performance on those outliers that deviate significantly from inliers but fail to capture outliers that are similar to inlier classes.

Thirdly, we examined the multi-task learning architecture, *supervised autoencoder* (SAE), for outlier detection. We found that combining the outlier detection objective as a classification objective with a self-supervised training objective yields better generalizing embeddings for the downstream outlier detection task.

Based on the partial deficiencies and advantages of each algorithm, we proposed the two new methods, an *adversarially trained autoencoder* (ATA), and an adversarial SAE (ASAE). With MLP leveraging outlier information within the classification objective and OCA explicitly learning an inlier representation, MLP and OCA are architecturally, fundamentally different, and succeed on opposing tasks. ATA leverages the outlier information to guide the inlier representation learning, which links the merits of MLP and OCA. Technically, ATA not only minimizes the inlier reconstruction error like OCA but also maximizes the reconstruction error for outliers, thereby allowing it to detect outliers that are conceptually related to inliers. Our results conclusively demonstrate the effectiveness of this method throughout various outlier detection and imbalanced classification experiments.

Following up on the idea of outlier reconstruction error maximization, we apply the same concept to SAE. Although the outlier detection performance does not significantly improve, we found that ASAE learns low dimensional embeddings in the latent space that are well-separated w.r.t. their class association. The separation demonstrates the effectiveness of the adversarial loss function, and allows for neural networks of reduced complexity for downstream classification tasks, such as the outlier

detection task.

In the next chapter, we reframe the outlier detection task as a one-vs-rest classification task, where the rest class includes observed and unobserved rest classes, and out-of-distribution data such as outliers and dataset shift. This setting is often referred to as open set recognition, and can be regarded as a generalized classification task that not only spans observed classes that can be learned via empirical risk minimization, but also unobserved classes and corruptions. Therefore, classifiers need to be able to not only correctly classify the known rest classes (closed set), but also to reject the unrestricted unknown (open set), to succeed on this complex task. Such classifiers are highly robust to unexpected conditions when deployed in the open world, a principal criterion for AI-safety.

We tackle open set recognition from the angle of supervised outlier detection, as it is capable of rejecting unknown conditions, i.e., the samples that deviate significantly from the inlier data generating process. In the next chapter, we extend ATA to learn a tight hull around the inlier data by utilizing the observed rest classes to guide the learning process, providing an effective solution to the second milestone. As we will prove, ATA and its extension have a bounded open space risk by design, which is not the case for the SAE variants.

Open Set Recognition

In this chapter, we tackle open set recognition (OSR), a highly complex and generalized version of one-vs-rest classification, via the supervised outlier detection method, ATA, developed in Ch. 3. OSR methods separate a fixed set of inlier classes from all the possibly existing rest classes, outliers and other corruptions, and therefore, match the complexity of real world deployments. In practice, models succeeding on this task provide tremendous robustness gains over traditional deep learning methods optimized via empirical risk minimization. After all, most of the AI accidents reported in Ch. 1 can be associated with out-of-distribution exposure at deployment time, and could have possibly been prevented by OSR methods. We argue that supervised outlier detection already provides the necessary means to detect unknown occurrences, making it a compelling direction for OSR. We explore the direction of applying ATA to OSR by modifying the architecture and reformulating the training objective to learn a hull around the inlier data that is as tight as possible, without compromising classification performance.

Firstly, we introduce the concept of OSR in this chapter and explain the intuition behind the robustness benefits of OSR methods in the open world. Secondly, we introduce the decoupled autoencoder (DAE) architecture, an advancement of ATA towards robust OSR. ATA separates the reconstruction error distribution of inliers and outliers and subsequently finds a threshold splitting inliers and outliers in reconstruction error space in a brute force fashion. In contrast, DAE actively minimizes the threshold without compromising the recall of inliers. Due to this training objective, the model learns a tighter hull around the inlier data, allowing the robust rejection of out-of-distribution data.

Thirdly, we elaborate on the differences between closed set classification and OSR, related to training objectives and robustness requirements. We then provide an upper bound on the open space risk of DAE, a crucial criterion for OSR models in open world deployments that is not met by the vast majority of OSR methods due to their architectural design. Next, we evaluate the method’s effectiveness over an extensive range of experiments, covering classification, outlier detection, and dataset shift. These benchmarks demonstrate DAE’s superiority over various state-of-the-art OSR methods. In particular, we show DAE’s robustness gains w.r.t. adversarial and improved calibration, unmatched by any of the baselines. Finally, we perform an ablation study on the multi-task optimization objective, and demonstrate that only the combination of each task leads to the desired tight hull around the inlier data.

This chapter is based on our publications [22, 99, 173], and provides a solution to the second milestone, specified in Sec. 1.5. The works are based on the idea to connect outlier detection and

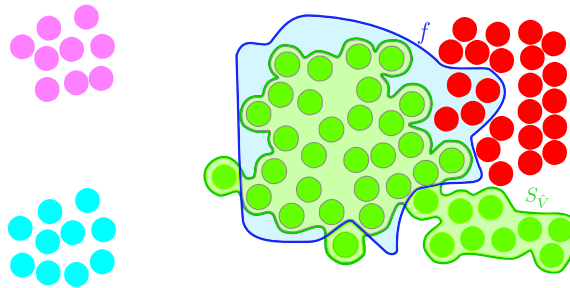


Figure 4.1: Conceptual illustration of OSR: The closed set S_V is visualized by the green area comprising all inliers. There are two different rest samples types: red points resemble a known rest class, whereas blue and pink samples reflect unobserved outliers. The objective is to learn an indicator function f (decision boundary shown as blue line), that filters inliers and rejects all rest samples. The open space risk $R_O(f)$ can be interpreted as the ratio of the blue area over the green area that is bounded by the decision boundary of f .

classification to OSR and resulted in the DAE architecture, both of which were contributed by Max Lübbering. Furthermore, the theoretical proofs of the existence of a bound on the open space risk and its estimation were carried out by Max Lübbering. Besides the theoretical work, Max Lübbering implemented the DAE architecture and the experimental setup, providing all the insights in the papers. Apart from the MiMo baseline [174], which was jointly implemented by Max Lübbering and his co-authors, Max Lübbering implemented all baselines. All authors provided extensive feedback during the implementation and evaluation of the DAE method, and helped in revising the final paper, which was written by Max Lübbering for most parts.

4.1 Introduction

Despite having achieved state-of-the-art classification results in diverse domains [175–177], DNNs are generally not well-equipped for real-world applications [18] and tend to fail when exposed to data from unseen distributions, as we have shown in Sec. 1.1.

This issue often goes unnoticed, as state-of-the-art classification results are primarily obtained in extremely controlled benchmark environments, with an inherent closed world assumption, raising the question of applicability to real-world scenarios [33–35, 178]. Specifically, DNNs tend to only generalize well within the concepts they were trained on, and tend to provide incorrect predictions with exaggerated confidence when exposed to samples from unseen distributions [18, 33–37], jeopardizing model robustness. As visualized in Fig. 4.2(b), the multi-layer perceptron (MLP) learns to separate the two half-moons and XOR circles, but fails to generalize to the uniform noise. Note that the MLP and the ensemble method MiMo [174] were trained via empirical risk minimization (ERM) to separate the two classes, so there is no preference to which class the outliers should be attributed. We would, however, expect a well-generalizing model to be uncertain about these samples and not assign them to one of the classes with high confidence.

In this work, we consider open set recognition (OSR) as the generalized version of one-vs-rest (OVR) classification [69], in which the rest samples not only stem from known classes, but also from different unknown sources of outlier generating processes [25, 179]. Therefore, the aforementioned deficiencies of DNNs also pose a severe threat to OSR. As suggested by [25], these deficiencies can be alleviated by framing the optimization problem in OSR as a combination of ERM and open space

risk minimization [25]. The open space is defined as the set of points that have at least distance d to any of the inlier samples, i.e., the closed space. The open space risk is then defined as the relative measure of the open space volume of false inliers, over the closed space volume classified as inliers (i.e., true inliers), as illustrated in Fig. 4.1. Consequently, minimizing the open space risk forces the model to learn a hull around the inliers, leading to increased model robustness.

The difference between ERM, and ERM in conjunction with open space risk minimization, is illustrated in Fig. 4.2. Due to its bounded open space risk, our proposed decoupled autoencoder (DAE) method learns a hull around the inlier data, whereas the MLP assigns an infinite space to the inlier class, resulting in infinite open space risk. In practice, this problem is often mitigated by the long-established background class setup [180], in which all rest classes are subsumed within a single background class [181, 182]. This approach can be an effective measure to learn a working OSR model, as illustrated in Fig. 4.2(h). The downside though, is that this approach is also highly data-intensive, often rendering it infeasible and error-prone, while still having an unbounded open space risk.

Similar as to supervised outlier detection, OSR generally deals with significant class imbalances. When the focus lies on filtering a single, narrow class of interest (COI), similar to the needle in the haystack problem, then this inlier class tends to be underrepresented. When the focus shifts towards outlier detection, for instance, in the case of computer virus detection, inlier samples outnumber instances of the rest class. Furthermore, only some classes of the problem domain are usually known for the set of rest classes (RCs), and outliers or dataset shifts are only witnessed at test time, making OSR a highly imbalanced, semi-supervised setting. Throughout the paper, we refer to COI samples as inliers, RC samples as rest samples, and samples of unseen rest classes as outliers.

To measure model robustness in isolation, we subdivide the OSR task into three disjunct sub-tasks by gradually increasing the scope of RC:

1. OVR classification task T_c : The model is evaluated on the COI and the RCs it was trained on.
2. Contextual outlier detection task T_o : Comprises the evaluation on COI and conceptually related RCs of T_c . In practice, the rest samples stem from the same dataset, but from RCs the model was not trained on. Consistent with the literature, we define samples originating from RCs in this case as contextual outliers, since they are possibly generated from a completely different underlying mechanism [42, 43, 183], yet contextually related.
3. Dataset shift task T_d : In this case, rest samples stem from a new dataset, equivalent to the evaluation approach in [34]. By extending the scope of RC in tasks T_o and T_d to rest classes unseen during training, aspects of semi-supervised outlier detection and robustness to dataset shift become predominant. In fact, tasks T_o and T_d can be seen as semi-supervised, one-class classification (OCC): In this semi-supervised setting, the model is only exposed to COI samples during training time, and is supposed to learn to reject samples that deviate from the COI representation, i.e., outliers [68, 70].

With our three-task experiment design, we are therefore able to bridge the gap between OVR and OCC within OSR, and precisely pinpoint the classification and robustness performance for each algorithm.

Due to the widespread application of OSR in safety-critical environments such as medical diagnosis [66, 184, 185], fraud detection [145, 186, 187], and intrusion detection [188, 189], the extension of

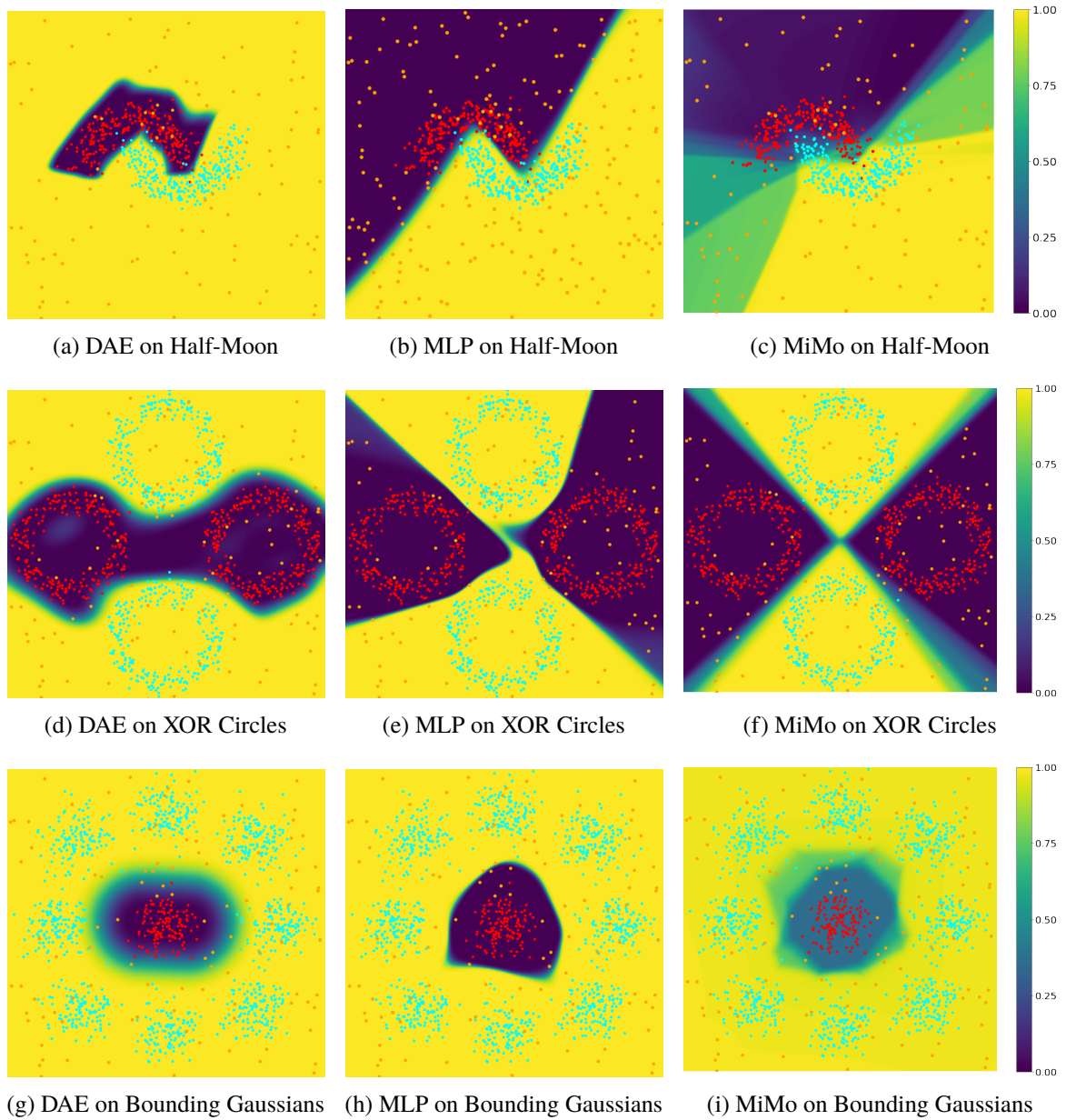


Figure 4.2: Class probabilities of DAE, MLP and MiMo [174] visualized as contours: In contrast to MiMo and MLP, DAE learns a hull around the red class of interest (COI) for the three datasets, enabling the method to not only separate the two inlier classes, but also to reject the unseen uniform noise. MLP and MiMo only wrap the inliers if the rest samples encourage such a decision boundary, as shown for the Bounding Gaussians dataset.

deep learning methods towards generalization and robustness with open space risk regularization is of supreme importance.

To this end, we propose the *decoupling autoencoder* (DAE) method, a novel, autoencoder-based architecture that learns a radial basis function (RBF) kernel, mapping the reconstruction error to class probabilities. The reconstruction error as a measure of outlierness is learned by a novel adversarial

loss function that separates inliers from rest samples in reconstruction error space, and is based on gradient ascent as suggested by us in [23] and Theorem 1. The inlier and outlier distributions are separated by a decision boundary that is optimized end-to-end to be as close as possible to the inlier distribution. Thus, observed and unobserved rest samples can be effectively rejected, resulting in an increased robustness. While the related ATA method employs gradient ascent to rest samples irrespective of their reconstruction error, and estimates the decision boundary offline in a brute-force fashion, the DAE loss function scales down the reconstruction error with increasing distance from the decision boundary. Therefore, samples close to the decision boundary are heavily optimized for better separation; hence the term "decoupling" in DAE. Furthermore, ATA only provides binary predictions therefore lacking rudimentary means of interpretation. In contrast, DAE yields subjective probability scores on the inlierness of a sample, substantially improving the model's interpretability, especially when deployed in safety-critical environments. The DAE architecture is sketched in Fig. 4.3.

Generally, our method combines the merits of MLPs succeeding on classification tasks, and autoencoder methods with their strong robustness. Additionally, we prove the existence of an upper bound on the open space risk for DAE and leverage this insight to actively minimize open space risk within DAE's loss function. Throughout a variety of experiments, we empirically show its benefits by outperforming the two most prominent OSR methods, C2AE [190] and OpenMax [58], outlier detection methods ATA [23] and *one-class autoencoder* (OCA), ensemble method MiMo [174], and one-vs-rest MLP.

With its added theoretical foundation and empirical verification, DAE is shown to be a promising candidate for deployment in safety-critical systems.

4.2 Related Work

The OVR classification strategy is often applied to binary models in order to extend them to multi-class classification [69]. Naturally, there is no necessity for OVR classification for DNNs, since they already support multi-class classification by design. However, there are common situations where OVR becomes relevant. Typically, in filtering tasks, in which we intend to filter a single positive class, we often only have access to the positive class, and a single negative class that subsumes all negative samples [191–193], rendering the problem to a binary OVR classification. As another example, if the goal is to filter normal samples from abnormal ones [179], there is also no class-wise distinction for the rest class. As motivated previously, vanilla MLPs are unsuitable in this case due to their infinite open space risk and consequential robustness deficiencies towards outliers [179]. While in modern architectures, researchers often try to circumvent this problem by subsuming all rest classes within a single background class [181, 182], this only reduces, but does not solve, the unbounded open space risk [179].

Due to the safety implications of an unbounded open space risk, there have been few attempts to bound it in DNNs. For instance, [58] and [194] leverage extreme value theory (EVT) to determine a compact abating probability model based on the deep features of the full network outputs. Noteworthy, both approaches are offline and are therefore not involved in training the network. The autoencoder-based approach C2AE, proposed by [190], also uses EVT to determine the decision boundary in reconstruction error space and requires at least two inlier classes. Other approaches try to bound open space risk by using tent activation functions [195]. They show that it increases robustness to adversarial attacks, while potentially compromising classification performance. In conclusion, as

stated in a recent survey, OSR is still largely unsolved within the deep learning domain [179].

In the related OCC setting, with its focus on outlier detection, DNN-based approaches have been researched from three angles: 1) Combining kernel methods [196] with DNN methods [63, 68, 197], 2) generative models (e.g., generative adversarial networks [198] or variational autoencoders [199]) based outlier detectors [66, 200, 201], and 3) based on (semi-) supervised autoencoders [23, 24, 64, 147, 148, 202]. Here, the key idea is to learn a representation of the inlier distribution, and subsequently, to estimate the outlierness of a sample via its reconstruction error. For a comprehensive study of outlier detection methods with different levels of supervision, we refer the reader to Ch. 1.

Other contributions focus on the calibration, as DNNs tend to provide wrong predictions with overly high confidence estimates for out-of-distribution samples [35, 77]. Since OVR classification incorporates outliers when extended to OCC, this issue is also prominent in OSR classification. Several methods to solve this have been proposed: Either by adding a calibration task to the model, which aligns it with the target probability distribution [203], or by incorporating diverse predictions from ensemble methods such as deep ensembles [35] and MiMo [174]. These methods combine multiple neural networks as weak classifiers, whose diverse outputs are aggregated to well-calibrated predictions; thus, compensating for overly confident predictions. Conversely, as pointed out by [204] and also supported by our findings, the diversity of the weak classifiers within the ensemble methods is not strong enough to generalize well to out-of-distribution samples. Instead, van Amersfoort et al. propose the kernel-based method DUQ, which learns centroids of classes in a lower-dimensional space [204]. Here, uncertainty is measured as a distance from the class centroids to out-of-distribution points.

In contrast to [204], our DAE method incorporates radial basis functions to estimate the outlierness of a sample via the distance to its reconstruction, and therefore does not require any centroid updating routines. Furthermore, ATA [23] optimizes the decision boundary w.r.t. F1 score in an offline, brute-force line search in reconstruction error space, which does not actively minimize the open space risk. For DAE, we designed a customized loss function that allows learning the decision boundary end-to-end and minimizes open space risk. Furthermore, it forces the classes to be more separated in reconstruction error space than ATA’s adversarial loss function.

4.3 Decoupling Autoencoders

Similar to existing autoencoder-based approaches for outlier detection [23, 64, 148], the decoupling autoencoder (DAE) method learns the outlierness of a sample via its reconstruction error. Existing approaches estimate the decision boundary via brute-force algorithms [23, 43] or learn the decision boundary via a subsequent downstream layer [24]. In contrast to this, DAE learns the decision boundary end-to-end, while optimizing for a pessimistic decision boundary that is most close to the inlier samples without compromising generalization performance. Thus, the decision boundary’s open space risk is actively minimized, a favorable setting in safety-critical systems.

From an architectural point of view, as displayed in Fig. 4.3, the network reconstructs a sample $\mathbf{x} \in \mathbb{R}^n$ using the autoencoder $\varphi(\mathbf{x}) = d(e(\mathbf{x}))$, consisting of an encoder $e(\cdot)$ and a decoder $d(\cdot)$. The reconstruction error e_{MSE} between the original and reconstructed sample $\hat{\mathbf{x}} \in \mathbb{R}^n$ computes to $e_{\text{MSE}}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{n} \sum_i^n (x_i - \hat{x}_i)^2$. The reconstruction error is mapped to the inlier probability via Gaussian $g : z \rightarrow e^{-\frac{z^2}{2\sigma^2}}$ with a mean of zero. Thus, the higher the reconstruction error, the smaller the inlier

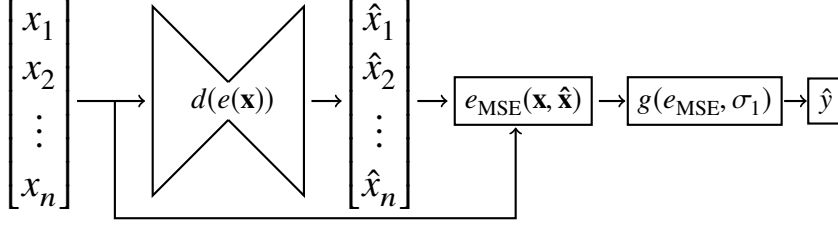


Figure 4.3: Decoupling Autoencoder (DAE) architecture: A joint architecture composed of an autoencoder $\varphi(\mathbf{x}) = d(e(\mathbf{x}))$ for sample reconstruction, a reconstruction error module e_{MSE} for outlieriness estimation, and an RBF kernel g with standard deviation σ_1 for classification. Thus, the entire network is given by $f(\mathbf{x}) = g(e_{\text{MSE}}(\mathbf{x}, d(e(\mathbf{x}))), \sigma_1)$.

probability. More formally, the full network f is given by

$$f(\mathbf{x}) = g(e_{\text{MSE}}(\mathbf{x}, d(e(\mathbf{x}))), \sigma_1). \quad (4.1)$$

Note that the standard deviation σ_1 is directly coupled with the decision boundary t via $t = \sqrt{-2\sigma_1^2 \ln \frac{1}{2}}$, as the threshold is fixed at the 0.5 level of function g , also shown in Fig 4.4.

During training, the network has three objectives:

- a) To minimize inlier reconstruction errors and maximize rest sample reconstruction errors, such that the inlier samples are easily distinguishable from rest samples within the one-dimensional reconstruction error space.
- b) To classify samples correctly based on the decision boundary t .
- c) To reduce the open space risk by minimizing the decision boundary t , such that the model is sensitive to unseen outliers.

The overall loss function \hat{L} incorporates these three training objectives by combining the adversarial loss function L_R , binary cross-entropy (BCE) classification loss $L_{\text{BCE}}(\hat{y}, y) = -[y \ln \hat{y} + (1-y) \ln(1-\hat{y})]$, and a regularizer term $|t|$, as follows:

$$\hat{L}(\mathbf{x}, \hat{\mathbf{x}}, y) = L_R(\mathbf{x}, \hat{\mathbf{x}}, y) + \lambda_2 L_{\text{BCE}}(f(\mathbf{x}), y) + \lambda_3 |t| \quad (4.2)$$

where y and \hat{y} denote the target label and the model's predicted COI probability of sample \mathbf{x} , respectively. Factors λ_2 and λ_3 scale the classification loss term and $|t|$ regularization.

The adversarial reconstruction loss L_R comprises the minimization and maximization of reconstruction errors w.r.t. inliers and rest samples, as defined by:

$$L_{R1}(\mathbf{x}, \hat{\mathbf{x}}, y) = L_{\text{MSE}}(\mathbf{x}, \hat{\mathbf{x}}) w_i(L_{\text{MSE}}(\mathbf{x}, \hat{\mathbf{x}})) \quad (4.3)$$

$$L_{R2}(\mathbf{x}, \hat{\mathbf{x}}, y) = L_{\text{MSE}}(\mathbf{x}, \hat{\mathbf{x}}) w_o(L_{\text{MSE}}(\mathbf{x}, \hat{\mathbf{x}})) \quad (4.4)$$

$$L_R(\mathbf{x}, \hat{\mathbf{x}}, y) = \begin{cases} \lambda_0 L_{R1}(\mathbf{x}, \hat{\mathbf{x}}, y), & y \in \text{inliers} \\ \lambda_1 L_{R2}(\mathbf{x}, \hat{\mathbf{x}}, y), & \text{otherwise,} \end{cases} \quad (4.5)$$

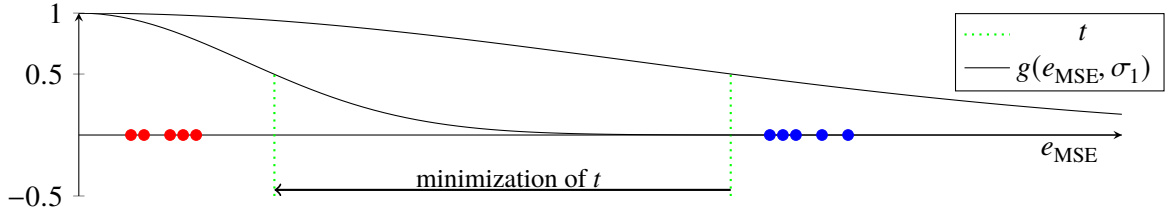


Figure 4.4: Combination of classification objective L_{BCE} and $\|t\|_1$ regularization to optimize for a decision boundary t , which is minimal, but does not compromise classification of inliers (red points) and rest samples (blue points)

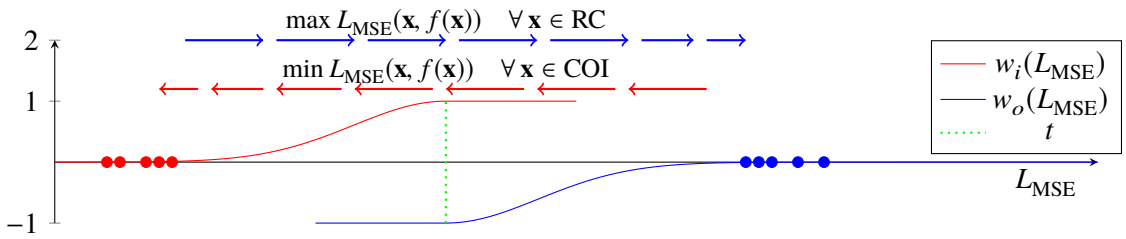


Figure 4.5: Intuition behind the reconstruction error objective L_R : Minimization/maximization of reconstruction losses for inliers (red) and rest samples (blue), respectively. Separation is achieved by weighting the reconstruction losses with w_i for inliers and w_o for rest samples. Therefore, samples on the wrong side of the decision boundary t are heavily optimized. With increasing distance from the decision boundary, optimization decelerates for correctly classified samples, as indicated by the length of the arrows.

where scaling factors $\lambda_0 \in \mathbb{R}^+$ and $\lambda_1 \in \mathbb{R}^+$ determine the minimization/maximization magnitude, respectively. Within L_R , the mean squared error $L_{\text{MSE}}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{n} \sum_i (x_i - \hat{x}_i)^2$ is weighted by $w_i : \mathbb{R} \rightarrow \mathbb{R}$ for inliers and by $w_o : \mathbb{R} \rightarrow \mathbb{R}$ for rest samples. These two functions, given by Eq. (4.6) and Eq. (4.7), push the reconstruction errors of inliers and rest samples away from the decision boundary t , towards the origin and ∞ , respectively, thereby providing a clear class separation:

$$w_i(l) = \begin{cases} 1 & l > t \\ e^{-\frac{(l-t)^2}{2\sigma_2^2}} & \text{otherwise} \end{cases} \quad (4.6)$$

$$w_o(l) = \begin{cases} -1 & l < t \\ -e^{-\frac{(l-t)^2}{2\sigma_2^2}} & \text{otherwise.} \end{cases} \quad (4.7)$$

Note that in both cases, the standard deviation σ_2 of the Gaussian is a hyperparameter and determines how far the two classes are being separated. σ_2 is not to be confused with σ_1 , which is coupled with the decision boundary t . The reconstruction error maximization of rest samples is achieved in Eq. (4.7) by the negation which is equal to flipping the loss gradients [23] and thus corresponds to gradient ascent. The reconstruction error objective L_R is conceptually visualized in Fig. 4.5.

Due to the spacious separation of the inlier set and rest sample set in reconstruction error space, there is a wide range of possible decision boundaries t . We argue that the best t is as close as possible

to the inlier set without compromising the classification performance, thus offering a reasonable trade-off between classification (T_c) and outlier detection (T_o) / dataset shift (T_d), while reducing the open space risk. The trade-off is modeled via the second and third addend of \hat{L} in Eq. (4.2). The $\|t\|_1$ regularizer minimizes the decision boundary t towards 0, eventually leading to an impractical classifier, always predicting RC independently of \mathbf{x} . This impractical solution is prevented by the classification loss term L_{BCE} acting as a stopping criterion, as visualized in Fig. 4.4. Note that the four scaling factors $\lambda_0, \dots, \lambda_3$ trade off these three objectives of reconstruction minimization/maximization, classification performance, and out-of-distribution robustness.

The combination of L_R and L_{BCE} also solves the vanishing gradient problem, which would occur for large e_{MSE} , due to the Gaussian output activation function. When solely minimizing the classification loss L_{BCE} jointly with the regularizer term, i.e., $\hat{L}^* = L_{\text{BCE}} + \|t\|_1$, we can show that

$$\lim_{e_{\text{MSE}}(\mathbf{x}, f(\mathbf{x})) \rightarrow \infty} \frac{\partial \hat{L}^*}{\partial \Theta} = 0, \quad (4.8)$$

where Θ are the network weights of autoencoder φ . The total derivative of \hat{L}^* computes to

$$\frac{\partial \hat{L}^*}{\partial \Theta} = \frac{\partial \hat{L}^*}{\partial f} \frac{\partial f}{\partial g} \frac{\partial g}{\partial e_{\text{MSE}}} \frac{\partial e_{\text{MSE}}}{\partial \varphi} \frac{\partial \varphi}{\partial \Theta}. \quad (4.9)$$

As e_{MSE} tends to infinity, Gaussian g becomes a horizontal line, resulting in gradients equal to 0:

$$\lim_{e_{\text{MSE}}(\mathbf{x}, f(\mathbf{x})) \rightarrow \infty} \frac{\partial g}{\partial e_{\text{MSE}}} = 0. \quad (4.10)$$

Thus, the expression in Eq. (4.8) zeros out, which is why the gradient updates become ineffective for inliers with large e_{MSE} . As L_R is independent of Gaussian g , it is not affected by this problem and enforces convergence by minimizing these inliers.

4.4 Classification Concern Conflicting with Robustness

During the training of DNNs, we minimize a surrogate loss function L , such as negative log-likelihood (NLL) instead of the non-differentiable 0-1 loss, over a given empirical data distribution $\hat{p}_{\text{data}}(\mathbf{x}, y)$, as the true $p_{\text{data}}(\mathbf{x}, y)$ is unknown [17]. As shown in Sec. 1.1, this corresponds to minimizing the empirical risk, which is given by $\mathbb{E}_{\mathbf{x}, y \sim \hat{p}_{\text{data}}(\mathbf{x}, y)}[L(f(\mathbf{x}; \Theta), y)] = \frac{1}{N} \sum_{i=1}^N L(f(\mathbf{x}^{(i)}; \Theta), y^{(i)})$, where N is the training set size, and f the model with parameters Θ . This procedure optimizes for a discriminating function which correctly separates the classes in the training set, i.e., optimization for classification performance. Under the assumption that the empirical data generating distribution $\hat{p}_{\text{data}}(\mathbf{x}, y)$ is similar to the true data generating distribution $p_{\text{data}}(\mathbf{x}, y)$, the discriminatory function will generalize to unseen data $\mathbf{x}, y \sim p_{\text{data}}(\mathbf{x}, y)$ within the problem domain.

A problem arises when the model is exposed to samples that are highly unlikely, according to $\hat{p}_{\text{data}}(\mathbf{x}, y)$, i.e., outliers because the model was not optimized w.r.t. such samples. This issue is visualized in Fig. 4.2 for MLP and MiMo on the Half-Moon and XOR Circles datasets. Both methods learn to separate the observed data (i.e., $\hat{p}_{\text{data}}(\mathbf{x}, y)$), and when we consider only the training data (red and blue samples), the learned model indeed minimizes the empirical risk. However, for out-of-distribution data (orange points), we would like to observe high uncertainty. Since this is not reflected in the

training objective, the model often predicts one of the two classes with high confidence, even though the out-of-distribution data cannot be attributed to either of the two classes [33].

A simple solution would be to facilitate reconstructive representation learning with, e.g., autoencoders, by forcing the reconstruction error to capture the outlieriness of a sample. As shown in Fig. 4.2 for DAE and in Fig. 4.6 for ATA, each model has learned a representation of COI, and rejects any major deviation from it as RC. While this indeed increases the robustness [23, 24], it can also harm the classification performance since a representation for all input features needs to be learned [204]. In fact, autoencoders aim to learn the natural manifold of the inlier data, which only exists under the manifold assumption introduced in Sec. 1.3. When the data lives on a highly complex manifold (see our related discussion on the challenges of encoding toxicities in online communication in Sec. 6.1.4), or the data is corrupted by uninformative, non-causal features, there can be a diminishing effect on the autoencoder’s classification performance. Models trained within the ERM framework do not suffer from this problem, as the feature extractor would neglect these features [204].

In conclusion, there is a trade-off between classification performance and robustness to outliers/dataset shift, which DAE aims to alleviate within the OSR framework.

4.5 Achieving Bounded Open Set Recognition with Autoencoders

In recent years, novel deep learning algorithms have advanced the state-of-the-art in many classification tasks. However, as noted in Sec. 4.1 and Sec. 1.1, it has also been shown that these algorithms, when solely optimized for empirical risk, often give wrong predictions with high confidence when exposed to dataset shift and outliers. In this section, we formalize this issue in line with [25], and prove that our approach has an upper bound on the open space risk, a primary criterion for robust OSR.

OSR was first defined by [25], was recently surveyed by [39, 179], and is still a largely unsolved topic within the deep learning domain [25, 58, 190]. OSR formalizes the problem of distinguishing a class of interest (i.e., samples originating from an observed set of classes) from samples derived from, e.g., outlier generating processes, dataset shifts, or other possibly unobserved but related classes. As deep learning classifiers are generally trained based on ERM by leveraging a surrogate loss such as cross-entropy, they only learn to differentiate the observed classes. This can be viewed as closed set classification, which is illustrated in Fig. 4.2: the MLP successfully learns to distinguish the two half-moons resembling the closed set. A problem arises when we zoom out of the problem domain and consider samples from the open set (i.e., outside of the closed set), then these samples that are far away from the closed set are still assigned to one of the two half-moons.

As a solution, OSR proposes an indicator function f over input space \mathbb{X} , that maps inliers to 1 and rest samples to 0. Partially following the notation of [25], let \hat{V} be the COI and $S_{\hat{V}} = \{\mathbf{x} \in \mathbb{X} \mid \min_{\mathbf{s} \in \hat{V}} \|\mathbf{x} - \mathbf{s}\| < d\}$ be the corresponding closed set, i.e., the set of all points within \mathbb{X} that are in d proximity to at least one of the inlier samples $\mathbf{s} \in \hat{V}$. Let $\mathcal{O} = \mathbb{X} - S_{\hat{V}}$ be the open space, then the open space risk is defined as

$$R_{\mathcal{O}}(f) = \frac{\int_{\mathcal{O}} f(\mathbf{x}) d\mathbf{x}}{\int_{S_{\hat{V}}} f(\mathbf{x}) d\mathbf{x}}, \quad (4.11)$$

which yields the ratio of the false inlier area over the true inlier area. Thus, $R_{\mathcal{O}}$ can be minimized by reducing the volume of the indicator function f . In Fig. 4.1, $R_{\mathcal{O}}$ can be interpreted as the ratio of the blue area (positive region of f outside $S_{\hat{V}}$, i.e., false positives) over the area of correctly predicted

inliers in $S_{\mathcal{V}}$ (i.e., true positives).

Note that a trivial solution for $R_{\mathcal{O}}$ minimization could be achieved by predicting all samples (except for one true inlier to prevent division by zero) as RC independently of their true class, i.e., resulting in a volume of 0 of the indicator function f over the open space. That is why it is crucial to counteract this solution by framing OSR as a two-fold problem with the two objectives: a) Minimization of $R_{\mathcal{O}}(f)$, and b) ERM as regularization.

As shown by the decision boundary of the MLP in Fig. 4.2 and denoted in prior research, vanilla MLPs do not provide an upper bound on the open space risk, and in practice, are generally unbounded [25, 58, 195]. Therefore, we have to turn towards deep learning architectures, such as autoencoders, that are capable of learning manifolds on the input space, and by design, learn a hull around the inliers:

Lemma 2. *Any autoencoder with saturating activation functions (e.g., sigmoid) within at least one of its layers and a reconstruction error output module acting as a manifold learner, has a bounded open space risk $R_{\mathcal{O}}$.*

Proof. Let $f(\mathbf{x}) = \mathbb{1}(|\varphi(\mathbf{x}) - \mathbf{x}| < d)$ be the recognition function. $f(\mathbf{x})$ is an indicator function, which maps the reconstruction error function of autoencoder $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ to $\{0, 1\}$ based on threshold $d \in \mathbb{R}$.

Let φ comprise at least one layer with an activation function that is saturating towards both tails (here, sigmoid). Assuming layer $\varphi^{(s)}$ to be the last layer with sigmoid activation and $\varphi^{(s)}$ to have m neurons, then the image this layer maps to is fixed within hypercube $(0, 1)^m$. Therefore, the image of all subsequent layers $\varphi^{(i)}$, $\forall i > s$, is also fixed. It follows that $\lim_{\mathbf{x} \in \mathbb{R}^n \rightarrow \infty} |\varphi(\mathbf{x}) - \mathbf{x}| = \infty$, as the image of $\varphi(\mathbf{x})$ is bounded. In conclusion, when starting from a sample classified as an inlier and moving in any fixed direction in feature space, then the reconstruction error will approach infinity. This forces us to cross the decision boundary of the recognition function f , proving the existence of a bound on the open space risk. \square

Lemma 3. *The open space risk can be approximated solely based on the weights of the layers succeeding the last layer with saturating activations.*

Proof. As previously defined, let $f(\mathbf{x})$ be the recognition function based on the autoencoder φ with $\varphi^{(s)}$ being the last layer with a saturating activation function $a^{(s)}$, and $a^{(s)}$ be the corresponding activations. Neurons in the subsequent layers can have a monotonic, non-saturating activation function $a^{(s+i)}$, $\forall i > 0$ (such as ReLu). For simplicity, we assume every layer $\varphi^{(i)}$ is comprised of m neurons. Let $\varphi^{(s)} \in (\inf_{x^* \in \mathbb{R}} a^{(s)}(x^*), \sup_{x^* \in \mathbb{R}} a^{(s)}(x^*))^m$, then the supremum of activation $\alpha_j^{(s+1)}$ and neuron $\varphi_j^{(s+1)}$ can be bounded by the following inequalities, respectively:

$$\sup_{x^* \in \mathbb{R}} \alpha_j^{(s+1)}(x^*) \leq \sum_{i=0}^m |\vartheta_{i,j}^{(s+1)} \sup_{x^* \in \mathbb{R}} a^{(s)}(x^*)| + |b_j^{(s+1)}| \quad (4.12)$$

$$\sup_{\mathbf{x} \in \mathbb{X}} \varphi_j^{(s+1)}(\mathbf{x}) \leq a^{(s+1)}(\sup_{x^* \in \mathbb{R}} \alpha_j^{(s+1)}(x^*)), \quad (4.13)$$

where $\vartheta_{i,j}^{(k)}$ denotes the weight between neuron i of layer $k - 1$ to neuron j of layer k , and $\vartheta_{0,j}^{(k)}$ denotes the bias of neuron j in layer k . It follows for the subsequent layers $s + 1 + l$,

$$\sup_{\mathbf{x} \in \mathbb{X}} \varphi_j^{(s+1+l)}(\mathbf{x}) \leq a^{(s+1+l)}(\sup_{x^* \in \mathbb{R}} \alpha_j^{(s+1+l)}(x^*)) \quad (4.14)$$

Therefore, the image of φ can be bounded by a hypercube with its center at the origin. It follows that the recognition function's open space $\int_{\mathcal{O}} f(\mathbf{x}) d\mathbf{x}$ can be approximated as the union of the set of points that are within the hypercube, and those that are in less than d proximity to the hypercube. \square

Lemma 2 and Lemma 3 have multiple practical implications. Firstly, our autoencoder methods are capable of bounding the open space risk and are therefore by design superior to MLP-based architectures in the OSR setting. Secondly, approximating the open space risk enables us to filter models with higher robustness during the model selection process. Thirdly, given that the approximated bound is a hypercube with its center in the origin of the feature space, it is recommendable to perform feature transformation such that the inlier class is located close to the origin, thus allowing for a smaller hypercube. Furthermore, Lemma 2 depicts the dependency of the open space risk on the weights after the last layer with saturating activation functions. By regularizing the weights in conjunction with the centering of the inlier samples, it is possible to actively minimize the bound on the open space risk. While this idea is out-of-scope for this contribution, it is a promising future research direction. For instance, it would be possible to handcraft the connections in the first layer of φ such that the weights perform translation and scaling of the feature space.

4.6 Towards Adversarial Robustness and Local Stability

Adversarial perturbations offer an effective way of measuring a model's robustness locally, as well as globally. As initially described by [37], the idea of gradient-based adversarial attacks is to confuse a neural network f with parameters Θ by adding an imperceptible perturbation to the original sample \mathbf{x} with target y . The perturbation itself is not arbitrary, but maximizes the loss L of the network. A common methodology of calculating these adversarial examples is the fast gradient sign method (FGSM) [37], which determines the perturbation by taking the sign of the gradients w.r.t. the sample:

$$\eta = \epsilon \operatorname{sign}(\nabla_{\mathbf{x}} L(f(\mathbf{x}; \Theta), y)), \quad (4.15)$$

where scaling factor ϵ determines the volume of change. Note that since ϵ is fixed, the perturbation's volume is also fixed for all steps across models.

While in practice, adversarial examples are often used to improve model robustness via adversarial training [37], in this work, we use the FGSM framework for robustness estimation of trained models. By perturbing a sample in a step-wise adversarial fashion, the model confidence development can be tracked, which provides deep insights into local stability, and with an increasing number of steps, also into global robustness of the model. Technically, at a given step i , a sample $\mathbf{x}^{(i)}$ is perturbed according to

$$\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \epsilon \operatorname{sign}(\nabla_{\mathbf{x}^{(i-1)}} L(f(\mathbf{x}^{(i-1)}; \Theta), y)) \quad (4.16)$$

and the difference in confidence Δc_i at step i w.r.t. the original sample is defined as

$$\Delta c_i(\mathbf{x}^{(0)}) = f(\mathbf{x}^{(i)}) - f(\mathbf{x}^{(0)}), \quad (4.17)$$

where $\mathbf{x}^{(0)}$ denotes the original sample. By varying the step-size ϵ and the number of steps, the aforementioned local stability and global robustness can be easily estimated. Furthermore, there also

exists another adversarial robustness metric [74], which is defined as:

$$\Psi(\mathbf{x}^{(0)}) = \min_{\hat{\mathbf{x}} \in \{\mathbf{x}^{(i)}\}_{V_i}} \frac{1}{D_{KL}(f(\mathbf{x}^{(0)}), f(\hat{\mathbf{x}}))} \quad (4.18)$$

and computes the Kullback-Leibler divergence D_{KL} in the denominator as the relative entropy between the two confidence estimates. We did not consider this measure, however, as D_{KL} rapidly changes around (0, 0) and (1, 1), and small changes in this area therefore lead to overly pessimistic robustness scores.

Generally, we hypothesize that DAE, as a representative for an OSR architecture, is more robust than the MLP, as the latter one slices the spaces in discriminative hyperplanes w.r.t. the two classes. Therefore, we can expect a given dataset shift sample to be closer to a hyperplane than DAE’s hull which is learned directly around the inlier class. We would assume similar adversarial robustness between MLP and DAE for an inlier sample.

4.7 Experiments and Results

In this section, we discuss the experiment setup and compare the performance of DAE to the aforementioned baselines. The approaches are compared on an algorithmic level by applying nested cross-validation (CV) [121, 205]. The best models are selected by the highest *area under the precision-recall curve* (AUPR) score, and are reported along with *area under the receiver operating characteristics* (AUROC) and F1 score with respective confidence measurements.

4.7.1 Selected Baselines

Since OSR touches upon multiple machine learning areas such as binary classification and outlier detection, we decided to consider the two most-prominent methods published under the OSR framework, and seven potent baselines from the outlier detection, OVR classification, and ensembling domains.

OpenMax: This is an offline OSR method that replaces the softmax layer within a fully trained multi-class DNN [58]. It applies extreme value theory to the network activations, which yields a final layer that outputs a probability score for the outlierness of a sample, and closed-set probability scores. While the method’s theoretical foundations are sound, it assumes that outlier activations differ significantly from inlier activations. In practice, this requirement is not always fulfilled [204], leading to robustness scores similar to those of the SoftMax baseline. By design, this method is limited to OSR problems with multiple COIs.

C2AE: This OSR approach [190] is derived from OCA which is used in outlier detection. The C2AE autoencoder is trained in a two-step fashion: 1) During closed set training, the encoder is trained jointly with a downstream classifier to perform closed set classification. 2) For decoder training, the latent vector is conditioned on an inlier-class-specific vector. During inference, it is assumed that an inlier has a lower reconstruction error compared to rest samples when the inlier’s latent vector is conditioned on the respective inlier-class-specific vector. Similar to OCA, this requires rest samples to be uncorrelated with inlier samples. By design, this method is limited to OSR problems with multiple COIs, and requires #closed-set classes many inference steps for each sample.

OCA: Classic semi-supervised, autoencoder-based method from the outlier and novelty detection domain that is trained to reconstruct inliers. The reconstruction error is assumed to be lower for inliers

than for outliers, thus rendering the reconstruction error predictive of the inlierness of a sample. As shown by [99], this method requires rest samples to be true outliers, as rest samples correlated with inliers tend to get reconstructed accurately.

ATA: This is a recent autoencoder-based method from supervised outlier detection, which, in contrast to OCA, actively maximizes the reconstruction error of rest samples [23]. Therefore, rest samples that are correlated with the COI also get maximized, thus alleviating OCA's aforementioned deficiencies.

MLP: OVR classification DNN, which subsumes the set of rest classes within a single background class [180–182]. This binary neural network requires the rest samples to wrap the COI in feature space, such that the model learns a decision boundary hull around the COI, making this approach highly data-intensive.

SoftMax: This DNN has a softmax layer for inlier probabilities as its final layer. The outlierness of a sample is estimated offline by the entropy of the sample's predicted inlier class probabilities (softmax output). Since the softmax predictions are generally overly confident for outlier data [33], the utility of this method is limited. By design, this method is restricted to problems with multiple COIs.

MiMo: This ensemble method [174] combines several weak classifiers into a single DNN by weight sharing, making this method more efficient compared to traditional ensemble DNNs. Similar to the OVR setup in MLP, this baseline subsumes non-COI classes into a single rest class. As shown by [35], ensemble methods can yield accurate predictive uncertainty estimates, which could ultimately improve OSR performance.

4.7.2 Evaluation Approach

Evaluating OSR models w.r.t. classification performance and robustness towards outliers and dataset shifts, is not straightforward. Firstly, the OSR setting is often highly imbalanced with observed rest samples significantly outnumbering the COI samples, whereas outliers/corruptions generally have a low prevalence in practice. Secondly, depending on the application's deployment domain, the focus between precision and recall can shift: For instance, in medical diagnosis systems, recall is often of the uttermost importance, whereas precision is generally favorable in equity trading precision. Due to this, threshold-dependent metrics such as F1 score or accuracy can be misleading and fail to capture the big picture.

As outlined in Sec. 1.2.1, we utilize the threshold-independent metrics AUPR and AUROC, as a viable solution to this issue since they evaluate the model at all possible thresholds [171].

Additionally, the F1 score is also reported, showing if each algorithm can learn reasonable decision boundaries, especially w.r.t. the classification task T_c . In contrast to the formerly mentioned AUROC and AUPR metrics, this metric evaluates the model at a fixed 0.5 threshold level, which is reasonable for task T_c , but is less conclusive for tasks T_o and T_d . Similar to AUPR, the F1 score metric is also affected by the base rate of the positive class.

Furthermore, the algorithms are evaluated based on the correctness of their subjective uncertainty estimation (calibration), in terms of the class-wise expected calibration error (CECE) [77–79] and Brier score [206]. While calibration is a crucial criterion for the trustworthiness of machine learning models, it is noteworthy that it is an orthogonal concern to model accuracy, e.g., in the trivial case, it is possible for a uniformly random classifier to be perfectly calibrated on a balanced dataset but still be inaccurate [35].

The CECE metric is defined as

$$\text{bin-CE}_{i,j} = |y_j(B_{i,j}) - \hat{p}_j(B_{i,j})| \quad (4.19)$$

$$\text{CECE} = \frac{1}{K} \sum_{j=1}^K \sum_{i=1}^m \frac{|B_{i,j}|}{N} \text{bin-CE}_{i,j} \quad (4.20)$$

where parameters K, m, N denote the number of classes, number of bins and dataset size, respectively. The set $B_{i,j}$ contains the samples whose confidence prediction w.r.t. class j falls into the i^{th} bin. The actual ratio of class j samples and average predicted confidence of samples in the bin is denoted by $y(B_{i,j})$ and $\hat{p}_j(B_{i,j})$, respectively. Therefore, $\text{bin-CE}_{i,j}$ denotes the calibration error for class j within bin i , and CECE is computed as the weighted average over all $\text{bin-CE}_{i,j}$.

The Brier score for binary classification is defined as

$$BS = \frac{1}{N} \sum_{i=1}^N (y_i - f(\mathbf{x}_i; \Theta))^2, \quad (4.21)$$

where N is the number of samples, y_i is the target of sample \mathbf{x}_i and $f(\mathbf{x}_i; \Theta)$ is the respective prediction (probability) of the model.

For a comprehensive robustness study, we also evaluated the algorithms w.r.t. their robustness to adversarial attacks and the related concern of local stability via the difference in confidence Δc_i (see Eq. (4.17)). As part of this, we empirically determined that a perturbation scaling factor $\epsilon = 0.001$ and $\#steps = 300$ captures both adversarial robustness and local stability. While, for MLP, it is straightforward to calculate the sample’s perturbation w.r.t. the binary cross-entropy loss, DAE’s training loss yields gradients that are almost 0 for small and large reconstruction errors, due to the Gaussian’s flatness at its mean and limits. Similar to Eq. (4.8), this results in dead updates within FGSM. As a solution, we calculate the gradients directly on the reconstruction error e_{MSE} .

4.7.3 Datasets

To evaluate the models on OSR with its classification subtask T_c and the more challenging subtasks of contextual outlier detection T_o and dataset shift T_d , we extended four textual datasets and three image datasets, some of which we already leveraged for supervised outlier detection in Ch. 3:

Reuters dataset: This multi-label dataset is a standard benchmark for document classification and outlier detection, which contains 10788 news documents from 90 different categories published by the news outlet Reuters. Since multi-label classification is out of the scope of this work, we only consider documents that are assigned to a single class.

ATIS dataset: This dataset comprises 5871 transcribed queries that passengers requested to the air travel information system (ATIS) for flight related information. These queries were grouped into 17 categories.

Newsgroups dataset: This dataset contains 18,000 newsgroup posts from 20 different topics and is a standard dataset for text classification and text clustering.

TREC dataset: A question classification dataset containing 5500 questions not limited to any particular topic domain [207]. This makes the dataset compelling for dataset shift evaluation.

MNIST dataset: An image classification dataset containing 70,000 images of handwritten digits

	ATIS		Reuters		Newsgroups		MNIST ₇		MNIST ₂₇		FMNIST ₇		FMNIST ₃₇		FMNIST _{0,1,2,3,7}	
	Inlier Rest		Inlier Rest		Inlier Rest		Inlier Rest		Inlier Rest		Inlier Rest		Inlier Rest		Inlier	Rest
S_t/S_c	flight	a_c	acq, earn	r_c	sci.space	n_c	7	m_c^7	2, 7	$m_c^{2,7}$	sneaker	f_c^7	dress, sneaker	$f_c^{3,7}$	t-shirt, pants, pullover, dress, sneaker	$f_c^{0,1,2,3,7}$
S_o	flight	a_o	acq, earn	r_o	sci.space	n_o	7	m_o	2, 7	m_o	sneaker	f_o	dress, sneaker	f_o	t-shirt, pants, pullover, dress, sneaker	f_o
S_{d1}	flight	t_d	acq, earn	t_d	sci.space	t_d	7	f_d	2, 7	f_o	sneaker	m_d	dress, sneaker	m_d	t-shirt, pants, pullover, dress, sneaker	m_d
S_{d2}	flight	r_d	acq, earn	a_d	sci.space	a_d	7	e_d	2, 7	e_d	sneaker	e_d	dress, sneaker	e_d	t-shirt, pants, pullover, dress, sneaker	e_d
S_{d3}	flight	n_d	acq, earn	n_d	sci.space	r_d	-	-	-	-	-	-	-	-	-	-

Table 4.1: Class assignment within training split S_t and evaluation splits S_c, S_o, S_{d1}, S_{d2} and S_{d3} for the text and image classification datasets, in accordance with [99]: Splits S_c and S_o are representative of tasks T_c and T_o , respectively; splits S_{d1}, S_{d2} , and S_{d3} of T_d . Mapping of rest classes specified in Table 4.2.

[208]. Each of the 28x28 gray-scale images shows a single digit (0-9).

FMNIST dataset: An image classification dataset comprising 70,000 fashion items [209]. Each of the 28x28 gray-scale images shows one of the ten fashion items: t-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, or ankle boot.

EMNIST-letter dataset: A dataset for image classification providing 145,600 grayscale images of alphabetic characters. Similar to TREC, we use this dataset solely for evaluation.

Due to the high number of different categories and their large size, each dataset is a viable benchmark dataset for OSR model evaluation. As shown in Table 4.1, we evaluated DAE and the baselines in seven different experiment setups. For each experiment setup, there is a single train split S_t and up to five test splits. All splits share the same set of inlier classes, which, depending on the derived dataset, vary from a narrow to a broader range of topics. The rest class covers a wide range of topics with an increasing scope from T_c, T_o to T_d . Specifically, classification split S_c shares all training classes, and therefore, resembles classic OVR classification T_c . Split S_o increases the scope by incorporating contextual outliers from the same dataset, as defined by the outlier detection task T_o . Finally, S_{d1}, S_{d2} , and S_{d3} have a maximum RC scope, by providing rest samples from a completely new dataset, representative of dataset shift task T_d . Concerning the preprocessing of the datasets, we vectorized the samples of the textual datasets as pooled 100-dimensional dense Glove embeddings [156], and z-transformed the image samples.

4.7.4 Results

To benchmark DAE against the aforementioned baselines, we train all models with approximately analogical complexity in terms of the number of trainable parameters. For text classification, the models MiMo, OpenMax, SoftMax, and MLP have three hidden layers of sizes 50, 25, and 12. The autoencoder-based approaches have three hidden layers of size 50, 25, and 12 for the encoder and the decoder in reverse order. For image classification, OpenMax, SoftMax and MLP have hidden layers of sizes 410, 256, 128, 64, and 43, while MiMo has hidden layer sizes of 120, 32 and 16. The autoencoder-based image classifiers have hidden layers of size 256, 128 for the encoder and the decoder in reverse order. C2AE has additional classification downstream layers of sizes 128, 32, 5. MiMo has five ensemble components and, therefore, an input size five times the input size of the other approaches. All neurons have sigmoid activation functions.

Within the nested CV, we performed a hyperparameter search concerning lr , *balanced sampling*, and *weight decay*, for all approaches. Specifically for DAE, we optimized for the initial decision boundary t and σ_2 and the loss scaling factors λ_i as defined in Eq. 4.2 and Eq. 4.7. ATA was optimized

Dataset	Abbr.	Rest labels
Reuters	r_c	carcass, cotton, cpi, crude, gnp, heat, housing, interest, ipi, jobs, livestock, lumber, money-fx, money-supply, nat-gas, oilseed, orange, pet-chem, reserves, retail, rubber, ship, tin, wpi, wpi
	r_o	alum, bop, cocoa, coconut, coffee, copper, fuel, gas, gold, grain, ground-nut, income, iron-steel, lei, meal-feed, nickel, potato, rice, sugar, tea, veg-oil, zinc
	r_d	alum, cocoa, coffee, copper, cotton, cpi, gold, grain, interest, ipi, jobs, money-fx, money-supply, nat-gas, reserves, rubber, ship, sugar, tin, trade, yen
ATIS	a_c	airfare, airline, ground_service
	a_o	abbreviation, restriction, airport, quantity, meal, city, flight_no, ground_fare, flight_time, distance, aircraft, capacity
	a_d	abbreviation, aircraft, airport, capacity, city, distance, flight_no, flight_time, ground_fare, meal, quantity, restriction
News groups	n_c	comp.os.ms-windows.misc, misc.forsale, rec.sport.baseball, sci.crypt, sci.med, soc.religion.christian, talk.politics.guns, talk.politics.misc
	n_o	rec.sport.hockey, comp.sys.ibm.pc.hardware, talk.religion.misc, talk.politics.mideast, comp.sys.mac.hardware, sci.electronics, alt.atheism, rec.motorcycles, rec.autos, comp.windows.x, comp.graphics
	n_d	alt.atheism, comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x, misc.forsale, rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, soc.religion.christian, talk.politics.guns, talk.politics.mideast, talk.politics.misc, talk.religion.misc
TREC	t_d	HUM, NUM, LOC, ABBR
MNIST	m_c^7	0, 1, 2, 3, 4, 9
	$m_c^{2,7}$	0, 1, 3, 4, 9
	m_o	5, 6, 8
	m_d	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
FMNIST	f_c^7	t-shirt, trouser, pullover, dress, coat, ankle boot
	$f_c^{3,7}$	t-shirt, trouser, pullover, coat, ankle boot
	$f_c^{0,1,2,3,7}$	coat, ankle boot
	f_o	sandal, shirt, bag
	f_d	t-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot
EMNIST-letter	e_d	a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z

Table 4.2: Specification of the rest classes for the different dataset splits in Table 4.1, derived from seven different textual and image-based classification datasets.

		AUROC Agg.		ATIS			NEWSGROUPS			MNIST ₇			FMNIST ₇		
		#best	#weak	AUROC ↑	AUPR ↑	F1 Score ↑	AUROC ↑	AUPR ↑	F1 Score ↑	AUROC ↑	AUPR ↑	F1 Score ↑	AUROC ↑	AUPR ↑	F1 Score ↑
S_c	DAE	0	0	98.6 ± 0.8	99.6 ± 0.2	97.9 ± 1.1	96.6 ± 0.6	86.0 ± 2.0	60.4 ± 26.4	99.3 ± 0.1	98.3 ± 0.2	96.8 ± 1.0	99.2 ± 0.3	96.8 ± 0.7	95.6 ± 1.1
	ATA	0	0	95.4 ± 1.0	98.9 ± 0.2	92.7 ± 1.4	94.8 ± 1.6	86.7 ± 2.1	74.7 ± 1.6	99.1 ± 0.2	98.7 ± 0.2	97.9 ± 0.1	98.9 ± 0.4	97.9 ± 0.4	96.6 ± 0.6
	OCA	0	2	72.0 ± 2.1	91.4 ± 1.1	64.1 ± 1.7	67.4 ± 1.8	28.4 ± 1.3	30.8 ± 3.2	94.9 ± 0.2	82.2 ± 0.5	70.1 ± 1.3	98.9 ± 0.1	93.3 ± 0.3	85.8 ± 0.6
	MLP	4	0	99.3 ± 0.1	99.8 ± 0.0	98.0 ± 0.4	97.5 ± 0.4	91.1 ± 0.9	79.6 ± 4.8	99.8 ± 0.1	99.3 ± 0.3	97.8 ± 0.1	99.9 ± 0.0	99.5 ± 0.1	96.5 ± 0.3
	MiMo	0	0	98.9 ± 0.1	99.8 ± 0.0	97.6 ± 0.4	97.1 ± 0.7	89.4 ± 1.3	81.9 ± 1.6	99.7 ± 0.0	98.7 ± 0.1	95.4 ± 0.5	99.8 ± 0.0	99.0 ± 0.1	95.9 ± 0.1
	BASE			50.0	81.9	31.0	50.0	11.4	9.3	50.0	14.6	11.3	50.0	14.3	11.1
S_o	DAE	1	0	89.3 ± 0.5	91.4 ± 0.2	84.1 ± 1.4	94.6 ± 1.2	50.3 ± 4.0	16.3 ± 9.6	99.3 ± 0.1	98.7 ± 0.1	96.7 ± 0.9	97.6 ± 0.2	89.6 ± 0.3	55.9 ± 0.9
	ATA	0	0	87.4 ± 0.5	89.9 ± 0.5	85.6 ± 0.8	91.8 ± 1.8	59.8 ± 2.4	18.5 ± 0.9	99.1 ± 0.2	98.8 ± 0.2	96.7 ± 0.4	96.9 ± 0.3	88.4 ± 0.7	52.1 ± 3.7
	OCA	1	2	80.4 ± 0.5	81.9 ± 0.6	62.1 ± 1.5	63.5 ± 1.9	6.6 ± 0.8	8.9 ± 0.9	99.7 ± 0.0	98.9 ± 0.1	73.9 ± 1.8	98.1 ± 0.1	90.3 ± 0.4	79.0 ± 0.5
	MLP	2	1	85.3 ± 1.7	86.2 ± 1.7	85.3 ± 0.8	95.6 ± 0.6	59.6 ± 3.3	25.0 ± 5.1	99.8 ± 0.0	98.7 ± 0.3	96.8 ± 0.1	90.8 ± 1.6	45.0 ± 6.5	53.5 ± 1.4
	MiMo	0	0	87.6 ± 1.9	90.0 ± 1.9	85.1 ± 0.8	95.1 ± 1.0	58.7 ± 4.6	48.0 ± 2.9	99.7 ± 0.0	98.9 ± 0.0	95.4 ± 0.4	95.9 ± 0.3	78.1 ± 2.1	55.8 ± 0.7
	BASE			50.0	61.8	27.6	50.0	1.9	1.8	50.0	12.7	10.1	50.0	11.8	9.5
S_{d1}	DAE	1	0	99.0 ± 0.1	98.4 ± 0.3	42.3 ± 2.2	97.7 ± 1.8	95.5 ± 2.7	46.5 ± 24.7	99.3 ± 0.1	99.0 ± 0.2	99.0 ± 0.2	99.3 ± 0.4	99.1 ± 0.5	90.1 ± 1.6
	ATA	0	0	97.9 ± 1.8	95.3 ± 3.1	85.2 ± 11.0	97.2 ± 0.6	55.3 ± 1.8	70.3 ± 1.7	98.6 ± 0.1	98.5 ± 0.2	60.8 ± 11.2	98.4 ± 0.1	98.0 ± 0.2	53.3 ± 5.2
	OCA	3	0	97.9 ± 0.2	96.5 ± 0.2	65.8 ± 1.6	99.3 ± 0.6	98.9 ± 0.7	45.1 ± 7.2	99.9 ± 0.0	99.9 ± 0.0	73.9 ± 1.8	99.9 ± 0.0	99.9 ± 0.0	92.7 ± 0.8
	MLP	0	3	73.1 ± 8.2	37.4 ± 7.7	46.3 ± 3.3	91.5 ± 3.0	31.1 ± 5.5	41.9 ± 9.6	99.8 ± 0.0	99.5 ± 0.1	98.5 ± 0.2	86.8 ± 2.7	51.1 ± 5.8	62.2 ± 6.9
	MiMo	0	3	80.5 ± 1.1	39.1 ± 1.5	44.3 ± 0.9	84.6 ± 1.7	26.7 ± 4.9	28.1 ± 2.3	99.8 ± 0.1	99.5 ± 0.1	97.1 ± 0.5	94.0 ± 2.7	78.1 ± 11.5	66.1 ± 4.7
	BASE			50.0	20.9	14.7	50.0	6.1	5.4	50.0	22.6	15.6	50.0	21.9	15.2
S_{d2}	DAE	0	0	98.3 ± 0.2	98.1 ± 0.2	59.7 ± 3.6	97.1 ± 2.5	95.3 ± 3.1	35.9 ± 25.7	99.3 ± 0.1	98.7 ± 0.2	97.1 ± 0.4	99.4 ± 0.3	98.9 ± 0.6	83.4 ± 1.9
	ATA	1	1	99.0 ± 0.3	98.7 ± 0.5	92.7 ± 0.6	90.2 ± 2.0	38.8 ± 4.0	26.9 ± 3.2	98.6 ± 0.1	98.3 ± 0.2	77.1 ± 12.6	98.6 ± 0.1	97.6 ± 0.2	42.6 ± 7.2
	OCA	3	0	94.3 ± 0.4	94.4 ± 0.4	65.8 ± 1.6	99.4 ± 0.5	99.1 ± 0.6	45.1 ± 7.2	99.6 ± 0.1	97.5 ± 0.3	73.7 ± 1.9	99.9 ± 0.0	99.8 ± 0.1	92.7 ± 0.8
	MLP	0	3	82.0 ± 10.0	68.1 ± 16.3	64.5 ± 5.5	81.4 ± 11.1	20.3 ± 18.8	19.0 ± 5.9	99.4 ± 0.1	94.2 ± 1.9	93.2 ± 1.4	90.6 ± 1.7	44.3 ± 5.8	49.4 ± 7.4
	MiMo	0	1	95.7 ± 0.8	92.1 ± 1.6	74.0 ± 1.6	60.2 ± 3.9	9.3 ± 5.0	8.1 ± 0.1	99.4 ± 0.1	96.6 ± 0.5	90.5 ± 0.9	95.8 ± 1.6	74.1 ± 11.0	55.6 ± 4.1
	BASE			50.0	34.6	20.5	50.0	4.2	3.9	50.0	12.3	9.9	50.0	11.9	9.6
S_{d3}	DAE	1	0	96.1 ± 0.5	87.8 ± 1.8	8.0 ± 0.1	96.8 ± 1.4	93.4 ± 2.6	42.7 ± 16.9						
	ATA	0	0	93.6 ± 2.0	81.7 ± 5.1	45.3 ± 20.7	96.2 ± 1.0	78.7 ± 2.8	71.8 ± 2.1						
	OCA	1	0	92.3 ± 0.5	83.3 ± 0.9	65.8 ± 1.6	98.9 ± 0.7	98.2 ± 0.9	45.1 ± 7.2						
	MLP	0	1	69.9 ± 8.1	12.7 ± 12.9	9.2 ± 1.3	95.9 ± 2.2	75.0 ± 10.0	70.2 ± 10.1						
	MiMo	0	1	92.2 ± 1.7	54.4 ± 4.3	11.1 ± 0.9	88.0 ± 2.3	57.4 ± 11.5	54.1 ± 3.9						
	BASE			50.0	4.1	3.8	50.0	11.5	9.3						

Table 4.3: Performance of DAE and the baselines ATA, OCA, MLP, and MiMo, on splits S_c , S_o , S_{d1} , S_{d2} and S_{d3} of the textual and image datasets with a single COI. The AUROC results are aggregated in the first column for each split, counting best and weak performances. Specifically, when a model’s AUROC score drops at least 5 percentage points below the best AUROC score, the model is counted as weak performing. For reference, BASE denotes the performance of a random classifier that predicts COI with probability $p \sim U[0, 1]$. Metrics and confidence are measured in %. Baseline results on textual datasets adopted from [99]. Across all three subtasks of OSR, DAE and ATA yield the most robust results, while MLP and MiMo show a significant performance degradation on the dataset shift task, similar to OCA on the classification task.

w.r.t. *outlier weighting factor* and *bin range*. We found that across various experiments, all baselines showed the best performances when optimized with Adam [32].

Table 4.3 and Table 4.4 show the results for each task with the best performing approaches on each dataset highlighted in boldface. To make model robustness comparable, a model is counted as weak when its AUROC score drops at least 5 percentage points below the best performing model. These scores are highlighted in gray within the result tables. Each results table aggregates the best and weakly performing models in the first column. Since the class base rates fluctuate significantly across datasets and splits, using AUPR and F1 Score as base rate dependent metrics were not considered for the model robustness evaluation. Table 4.3 and Table 4.4 present the results on datasets with a single COI and multiple COIs, respectively.

On the classification task T_c , represented by split S_c , MLP and MiMo yield the best results on all datasets. DAE and ATA also provide competitive results on this task, but OCA, C2AE, OpenMax, and SoftMax fail on almost all S_c splits in terms of AUROC.

For the contextual outlier detection task T_o , we see that DAE and ATA outperform MLP and MiMo in terms of AUPR and AUROC scores on the multi-COI datasets. DAE and ATA provide similar results to MiMo and MLP on the single-COI datasets. As expected, the semi-supervised methods OCA, SoftMax, OpenMax, and C2AE do not achieve any performance gains.

4.7 Experiments and Results

		AUROC Agg.		REUTERS			MNIST _{2,7}			FMNIST _{3,7}			FMNIST _{0,1,2,3,7}		
		#best	#weak	AUROC \uparrow	AUPR \uparrow	F1 Score \uparrow	AUROC \uparrow	AUPR \uparrow	F1 Score \uparrow	AUROC \uparrow	AUPR \uparrow	F1 Score \uparrow	AUROC \uparrow	AUPR \uparrow	F1 Score \uparrow
S_c	DAE	0	0	99.2 \pm 0.3	99.7 \pm 0.1	99.0 \pm 0.1	99.0 \pm 0.1	97.8 \pm 0.1	96.8 \pm 0.4	97.5 \pm 0.3	93.2 \pm 0.9	91.2 \pm 0.7	97.1 \pm 0.2	98.6 \pm 0.1	95.9 \pm 0.1
	ATA	0	0	99.4 \pm 0.2	99.8 \pm 0.1	97.8 \pm 0.6	99.4 \pm 0.2	99.2 \pm 0.2	98.4 \pm 0.1	97.0 \pm 0.3	95.9 \pm 0.2	93.5 \pm 0.4	97.2 \pm 0.5	98.9 \pm 0.2	96.3 \pm 0.3
	OCA	0	4	78.3 \pm 5.3	92.7 \pm 2.2	73.5 \pm 3.8	73.8 \pm 1.9	58.6 \pm 2.5	34.9 \pm 3.1	91.3 \pm 0.2	82.4 \pm 1.1	73.0 \pm 0.5	81.2 \pm 0.6	91.7 \pm 0.4	76.5 \pm 1.2
	C2AE	0	4	83.3 \pm 0.7	94.6 \pm 0.3	66.2 \pm 2.3	79.7 \pm 1.5	65.8 \pm 1.7	53.2 \pm 0.3	90.7 \pm 0.6	80.8 \pm 1.2	64.3 \pm 3.1	82.6 \pm 0.4	92.7 \pm 0.3	83.8 \pm 0.8
	OpenMax	0	4	74.5 \pm 2.2	90.3 \pm 2.4	87.2 \pm 0.0	84.3 \pm 1.0	69.4 \pm 2.0	44.4 \pm 0.0	53.8 \pm 9.7	31.9 \pm 6.6	44.4 \pm 0.0	66.6 \pm 12.7	82.8 \pm 7.3	83.3 \pm 0.0
	SoftMax	0	4	74.4 \pm 2.1	90.3 \pm 2.4	87.2 \pm 0.0	84.3 \pm 1.0	69.4 \pm 2.0	44.4 \pm 0.0	56.2 \pm 11.8	34.5 \pm 7.3	44.5 \pm 0.0	63.0 \pm 2.0	75.6 \pm 1.0	83.3 \pm 0.0
	MLP	4	0	99.7 \pm 0.1	99.9 \pm 0.1	99.0 \pm 0.3	99.9 \pm 0.1	99.8 \pm 0.1	98.9 \pm 0.2	99.4 \pm 0.1	98.6 \pm 0.2	93.8 \pm 0.4	98.7 \pm 0.1	99.2 \pm 0.1	96.5 \pm 0.1
	MiMo	1	0	99.6 \pm 0.1	99.9 \pm 0.0	98.6 \pm 0.1	99.7 \pm 0.0	99.3 \pm 0.1	96.8 \pm 0.3	98.7 \pm 0.2	97.1 \pm 0.4	91.2 \pm 1.0	98.2 \pm 0.1	99.2 \pm 0.1	95.8 \pm 0.2
BASE			50.0	77.3	30.4	50.0	28.6	18.2	50.0	28.6	18.2	50.0	71.4	29.4	
S_o	DAE	2	0	98.9 \pm 0.2	99.2 \pm 0.3	97.8 \pm 0.2	98.7 \pm 0.1	97.1 \pm 0.2	81.2 \pm 3.0	95.2 \pm 0.2	87.4 \pm 0.3	64.3 \pm 1.2	89.2 \pm 0.3	84.4 \pm 0.4	67.9 \pm 0.8
	ATA	1	0	99.4 \pm 0.0	99.7 \pm 0.0	97.1 \pm 0.6	99.3 \pm 0.1	98.9 \pm 0.2	90.0 \pm 2.9	93.9 \pm 0.4	87.2 \pm 0.6	56.9 \pm 0.9	88.0 \pm 0.8	83.8 \pm 0.4	64.0 \pm 0.4
	OCA	0	2	75.1 \pm 4.1	86.1 \pm 2.7	70.8 \pm 3.7	95.4 \pm 1.1	90.8 \pm 2.2	36.4 \pm 3.4	92.6 \pm 0.3	78.7 \pm 1.5	71.3 \pm 0.7	83.4 \pm 0.3	72.9 \pm 0.8	69.3 \pm 0.8
	C2AE	0	2	76.6 \pm 1.4	87.0 \pm 0.8	64.8 \pm 2.1	95.6 \pm 0.9	91.0 \pm 1.8	67.5 \pm 2.7	87.6 \pm 0.5	66.6 \pm 1.7	57.2 \pm 3.4	84.8 \pm 1.2	78.5 \pm 2.2	70.3 \pm 3.2
	OpenMax	0	4	81.0 \pm 2.4	87.3 \pm 3.1	77.4 \pm 0.0	76.2 \pm 1.6	46.1 \pm 2.4	36.3 \pm 0.0	72.7 \pm 12.8	41.0 \pm 13.9	34.8 \pm 0.0	77.0 \pm 7.9	65.9 \pm 13.7	57.1 \pm 0.0
	SoftMax	0	4	81.0 \pm 2.3	87.3 \pm 3.1	77.4 \pm 0.0	76.2 \pm 1.6	46.1 \pm 2.4	36.3 \pm 0.0	66.1 \pm 12.3	34.4 \pm 14.3	34.8 \pm 0.1	77.4 \pm 1.1	54.4 \pm 2.7	57.1 \pm 0.0
	MLP	0	2	99.3 \pm 0.1	99.3 \pm 0.2	97.5 \pm 0.2	99.1 \pm 0.2	97.7 \pm 0.4	79.3 \pm 2.4	89.3 \pm 0.7	63.9 \pm 1.8	62.8 \pm 1.4	74.8 \pm 5.2	60.0 \pm 6.1	64.2 \pm 0.7
	MiMo	1	1	99.6 \pm 0.1	99.8 \pm 0.1	97.5 \pm 0.3	98.5 \pm 0.2	96.9 \pm 0.3	83.3 \pm 1.9	91.9 \pm 1.1	77.9 \pm 3.2	63.2 \pm 1.6	84.3 \pm 0.4	75.8 \pm 0.7	65.9 \pm 1.1
BASE			50.0	63.1	27.9	50.0	22.2	15.4	50.0	21.1	14.8	50.0	40.0	22.2	
S_{d1}	DAE	1	0	99.5 \pm 0.2	99.3 \pm 0.2	63.3 \pm 3.6	98.8 \pm 0.5	98.8 \pm 0.4	82.0 \pm 6.1	98.1 \pm 0.2	98.2 \pm 0.2	81.3 \pm 3.3	98.0 \pm 0.3	99.0 \pm 0.2	93.9 \pm 1.2
	ATA	0	0	97.0 \pm 0.7	96.2 \pm 0.8	64.4 \pm 1.1	98.9 \pm 0.3	99.0 \pm 0.2	65.4 \pm 2.6	96.1 \pm 0.2	96.5 \pm 0.2	63.2 \pm 1.2	94.7 \pm 1.5	97.5 \pm 0.7	76.2 \pm 0.9
	OCA	3	0	98.6 \pm 0.2	98.1 \pm 0.4	75.5 \pm 3.3	99.6 \pm 0.3	99.2 \pm 0.7	36.3 \pm 3.3	99.8 \pm 0.0	99.8 \pm 0.0	80.1 \pm 1.0	99.5 \pm 0.1	99.7 \pm 0.1	80.8 \pm 1.6
	C2AE	0	1	98.7 \pm 0.1	98.4 \pm 0.1	66.7 \pm 2.3	98.5 \pm 0.4	97.3 \pm 0.8	67.5 \pm 2.7	94.0 \pm 1.5	92.9 \pm 2.0	68.8 \pm 3.0	97.6 \pm 0.5	98.6 \pm 0.3	89.9 \pm 3.4
	OpenMax	0	4	75.0 \pm 4.9	70.4 \pm 5.1	45.0 \pm 0.0	74.3 \pm 1.7	56.1 \pm 3.5	53.3 \pm 0.0	84.5 \pm 16.9	80.0 \pm 22.3	52.9 \pm 0.1	83.7 \pm 10.8	84.6 \pm 10.6	73.7 \pm 0.0
	SoftMax	0	4	74.4 \pm 5.1	69.4 \pm 5.4	45.0 \pm 0.0	74.3 \pm 1.8	56.1 \pm 3.5	53.3 \pm 0.0	71.3 \pm 15.3	63.5 \pm 20.1	52.9 \pm 0.1	86.4 \pm 1.6	83.6 \pm 3.4	73.7 \pm 0.0
	MLP	0	3	90.1 \pm 1.8	68.0 \pm 6.3	67.0 \pm 1.8	84.7 \pm 3.3	67.2 \pm 4.2	58.3 \pm 2.0	94.9 \pm 1.3	90.3 \pm 2.6	83.7 \pm 2.3	66.6 \pm 6.5	69.5 \pm 6.0	75.7 \pm 1.2
	MiMo	0	4	92.8 \pm 2.1	81.9 \pm 4.6	59.6 \pm 3.1	93.0 \pm 1.8	91.0 \pm 1.2	67.5 \pm 8.6	90.6 \pm 3.5	87.1 \pm 5.8	73.4 \pm 4.7	80.9 \pm 1.9	85.5 \pm 2.8	77.0 \pm 0.6
BASE			50.0	29.0	18.4	50.0	36.4	21.1	50.0	35.9	20.9	50.0	58.3	26.9	
S_{d2}	DAE	1	0	99.3 \pm 0.1	98.9 \pm 0.3	49.9 \pm 7.3	97.1 \pm 0.6	86.2 \pm 3.1	76.5 \pm 4.8	98.4 \pm 0.2	97.5 \pm 0.3	78.1 \pm 3.5	97.8 \pm 0.3	98.1 \pm 0.2	86.8 \pm 2.6
	ATA	1	0	98.1 \pm 0.5	96.6 \pm 0.8	66.1 \pm 2.7	97.8 \pm 0.2	93.0 \pm 1.2	55.0 \pm 6.2	96.6 \pm 0.3	95.4 \pm 0.4	51.0 \pm 2.2	94.5 \pm 1.6	96.0 \pm 1.1	60.9 \pm 1.4
	OCA	2	0	98.4 \pm 0.3	97.7 \pm 0.6	75.5 \pm 3.3	88.7 \pm 2.8	56.4 \pm 9.5	32.1 \pm 4.5	99.8 \pm 0.0	99.6 \pm 0.1	80.1 \pm 1.0	99.4 \pm 0.1	99.3 \pm 0.2	80.7 \pm 1.6
	C2AE	0	0	98.5 \pm 0.1	98.0 \pm 0.2	66.7 \pm 2.3	97.2 \pm 0.2	91.2 \pm 0.5	66.7 \pm 2.7	97.1 \pm 0.5	93.6 \pm 1.3	68.8 \pm 3.3	98.6 \pm 0.2	98.5 \pm 0.3	89.3 \pm 4.1
	OpenMax	0	4	83.5 \pm 1.7	72.7 \pm 3.0	35.5 \pm 0.0	74.4 \pm 1.4	35.9 \pm 1.5	35.5 \pm 0.0	82.1 \pm 15.3	66.1 \pm 24.9	35.0 \pm 0.0	80.4 \pm 10.0	70.2 \pm 14.5	57.4 \pm 0.0
	SoftMax	0	4	83.3 \pm 1.7	72.6 \pm 2.9	35.5 \pm 0.0	74.4 \pm 1.4	35.9 \pm 1.5	35.5 \pm 0.0	71.5 \pm 13.5	48.9 \pm 24.4	35.1 \pm 0.1	81.7 \pm 2.0	61.8 \pm 3.9	57.4 \pm 0.0
	MLP	0	3	84.2 \pm 5.6	49.2 \pm 16.0	47.5 \pm 4.4	94.8 \pm 0.7	76.5 \pm 2.5	63.9 \pm 1.8	93.8 \pm 1.7	77.9 \pm 6.3	70.9 \pm 4.7	65.3 \pm 3.8	51.5 \pm 4.2	60.6 \pm 0.6
	MiMo	0	3	82.1 \pm 8.0	58.4 \pm 17.2	39.3 \pm 1.8	96.3 \pm 0.6	88.2 \pm 1.4	70.8 \pm 2.4	93.6 \pm 2.2	84.3 \pm 5.4	68.2 \pm 6.3	81.3 \pm 1.5	76.6 \pm 3.6	62.3 \pm 0.7
BASE			50.0	21.6	15.1	50.0	21.6	15.1	50.0	21.2	14.9	50.0	40.2	22.3	
S_{d3}	DAE	0	0	96.3 \pm 1.7	88.2 \pm 3.4	19.6 \pm 5.2									
	ATA	0	0	93.1 \pm 1.6	85.6 \pm 2.1	14.0 \pm 0.3									
	OCA	0	1	79.7 \pm 10.0	56.6 \pm 23.0	47.0 \pm 26.0									
	C2AE	0	1	89.4 \pm 0.6	80.4 \pm 0.9	66.6 \pm 2.3									
	OpenMax	0	1	90.5 \pm 2.0	69.5 \pm 6.2	11.7 \pm 0.0									
	SoftMax	0	1	90.6 \pm 1.9	69.6 \pm 6.2	11.7 \pm 0.0									
	MLP	1	0	97.1 \pm 1.1	67.6 \pm 13.3	27.6 \pm 4.4									
MiMo	0	0	95.2 \pm 4.3	86.6 \pm 10.5	15.8 \pm 3.9										
BASE			50.0	6.2	5.5										

Table 4.4: Performance of DAE and the baselines on the derived datasets with more than one COI. Across all tasks DAE and ATA show high robustness, whereas the remaining baselines perform poorly on at least a single task. Baseline results on the Reuters dataset partially adopted from [99].

Concerning the dataset shift task T_d , the autoencoder-based methods yield, by far, the strongest results, with DAE being the only method with a zero weak count and OCA providing the most top scores. In contrast, the performance of MLP and MiMo diminish further from T_o to T_d , while the results of OpenMax, C2AE, and SoftMax improve on T_d compared to T_o . On average, the autoencoder-based methods have a weak performance rate of 6%, whereas the remaining baselines have a weak performance rate of 80%, clearly depicting the superiority of the autoencoder-based methods on the task T_d .

Taking the architectural properties of each method into account, we can conclude the following: The OVR baselines MLP and MiMo require the one-vs-rest relationship to be reflected within the data, similar to the Bounding Gaussians dataset example in Fig. 4.2. Only in this case, the ERM objective encourages the construction of a hull around the inlier data, which generalizes to unseen rest classes. This fails, however, when the number of COI classes increases (see FMNIST_{0,1,2,3,7} results), as there is no data-inherent rest preference for unseen classes. This leads to poor robustness scores on the outlier detection and dataset shift task. Compliant with earlier research [33, 34], MLP and MiMo

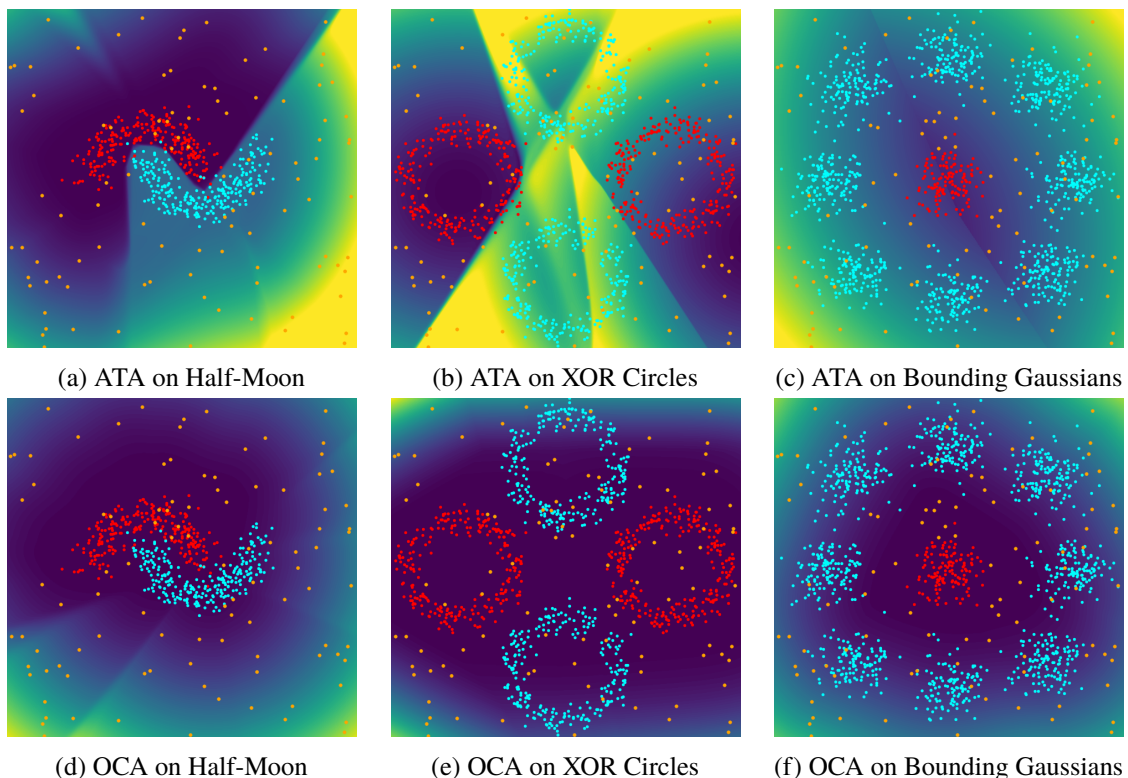


Figure 4.6: Reconstruction error landscape of ATA and OCA, visualized on the Half-Moon, XOR Circles, and Bounding Gaussians dataset, equivalent to Fig. 4.2. The color gradient from blue to yellow resembles an increase in the reconstruction error.

reveal the most severe robustness deficiencies when facing dataset shift. While, in practice, OSR is often approached by subsuming all the rest samples within a single background class, our results display that this is insufficient.

Conversely, semi-supervised OCA and C2AE are not able to learn the OVR relationship in the problem domain, indicating that inliers and rest samples within T_c and T_o are too correlated in features space. When the scope of OVR is extended to dataset shift, OCA outperforms all baselines, and C2AE becomes competitive. While the results for SoftMax and OpenMax express the same behavior, the underlying reasons are different: ERM has no intrinsic mechanism that prevents outliers from being mapped to inlier feature representations, a problem described as feature collapse, that is prevented via, e.g., two-sided Lipschitz constant regularization [204]. OpenMax and SoftMax both suffer from this effect, since both of them apply offline uncertainty estimates solely based on the activations. Anecdotally, we also replaced the sigmoid activation functions with RELU within OpenMax, since the network becomes piece-wise linear with the possibility of more expressive activations. This did not lead to any robustness improvements, however.

In contrast to the aforementioned baselines, DAE and ATA do not express any robustness deficiencies. In fact, they provide competitive results on all three OSR subtasks, showing that they are able to distinguish the two OVR classes in the problem domain, while also being highly robust to dataset shift. Nevertheless, we find that there is still a small trade-off between accuracy and robustness, which has

been reported in previous research for various deep learning methods [210, 211]. Both methods use an adversarial loss function that minimizes/maximizes inlier and rest sample reconstruction errors, respectively. Therefore, these methods resolve OCA’s issue of correlated rest and inlier samples within T_c and T_o . Additionally, due to the bounded open space risk, they suffer fewer remote artifact areas, which map outliers to inlier data, as seen for MLP and MiMo. While DAE and ATA are the most robust models, DAE is the best performing model in 7 cases, compared to ATA, which performs best in only 3. Since both methods mainly differ in terms of decision boundary estimation, the results suggest that DAE’s loss function, with its BCE term and t regularization, has a positive effect.

The robustness properties of DAE and the baselines can also be seen in Fig. 4.2 and Fig. 4.6. DAE, ATA, MiMo, and MLP are capable of separating the red inliers from the blue rest samples, however, in a fundamentally different way. While DAE and ATA learn a representation for the inlier class, MiMo and MLP learn a separating line, which does not generalize to the unobserved orange outliers. As shown in Fig. 4.2(h), the background class setup enables the ERM models MLP and MiMo to learn a proper hull around the inlier samples only if the observed rest samples bias the ERM towards such a decision boundary. Finally, OCA learns to separate the inliers from the orange outliers but passively minimizes the rest samples along with the inliers. This explains the poor classification performance of OCA on T_c and T_o , but high robustness to dataset shift.

Similar conclusions on the autoencoder-based methods can be drawn from Fig. 4.7, which displays samples from each split of the MNIST₇ and FMNIST_{3,7} datasets. DAE and ATA can reconstruct inliers and distort rest samples, resulting in a reconstruction error that is highly predictive of the inlierness of a sample. In contrast, OCA not only learns to reconstruct inlier samples, but also implicitly learns to reconstruct rest samples that originate from the same problem domain, explaining its low AUROC scores on T_c and T_o . Similarly, C2AE consistently reconstructs a sample as one of the two inlier classes so the overall reconstruction quality is much lower compared to the other two approaches. This could be caused by the joint encoder/downstream layer training which aims for classification instead of reconstruction. While the autoencoder-based methods have a bounded open space risk by design, the adversarial training within DAE and ATA forces the rest samples to be in the open space, far away from the decision boundary, as indicated by the rest sample distortions. Therefore, DAE and ATA are superior in classification settings T_c , compared to the semi-supervised autoencoder methods.

With adversarial robustness and local stability, there are two additional, crucial aspects of robustness, which can be measured by the change in model confidence after step-wise applying adversarial perturbations, as defined in Eq. (4.17). As shown in Fig. 4.8, both DAE and MLP are similarly stable when exposed to adversarially perturbed inliers of MNIST₇. Regarding rest samples, there is a substantial robustness gain from S_c to S_d on MNIST₇, with DAE being significantly more robust than the MLP. On FMNIST_{3,7}, the increased diversity of inlier samples diminishes the MLP’s adversarial robustness. This supports our presumption that the MLP’s recognition function has artifact areas far from the inliers that erroneously map to the inlier class. Conversely, the increase in inlier diversity enhances the inlier adversarial robustness of DAE. In theory, this forces DAE to learn a more voluminous decision boundary hull that is less susceptible to inlier perturbations.

As shown in Table 4.5, we also investigated the well-known problem of poorly calibrated neural networks on out-of-distribution examples after being trained via ERM [35, 174]. Specifically, we compared DAE to MLP and MiMo, to check if the DAE architecture improves model calibration. The results indicate that the three methods are similarly well-calibrated on task T_c in terms of Brier score and CECE. On tasks T_o and T_d , we found that each method provided poor calibration with such a high variance across datasets, that we omitted these inconclusive results. Instead, we analysed the similarity

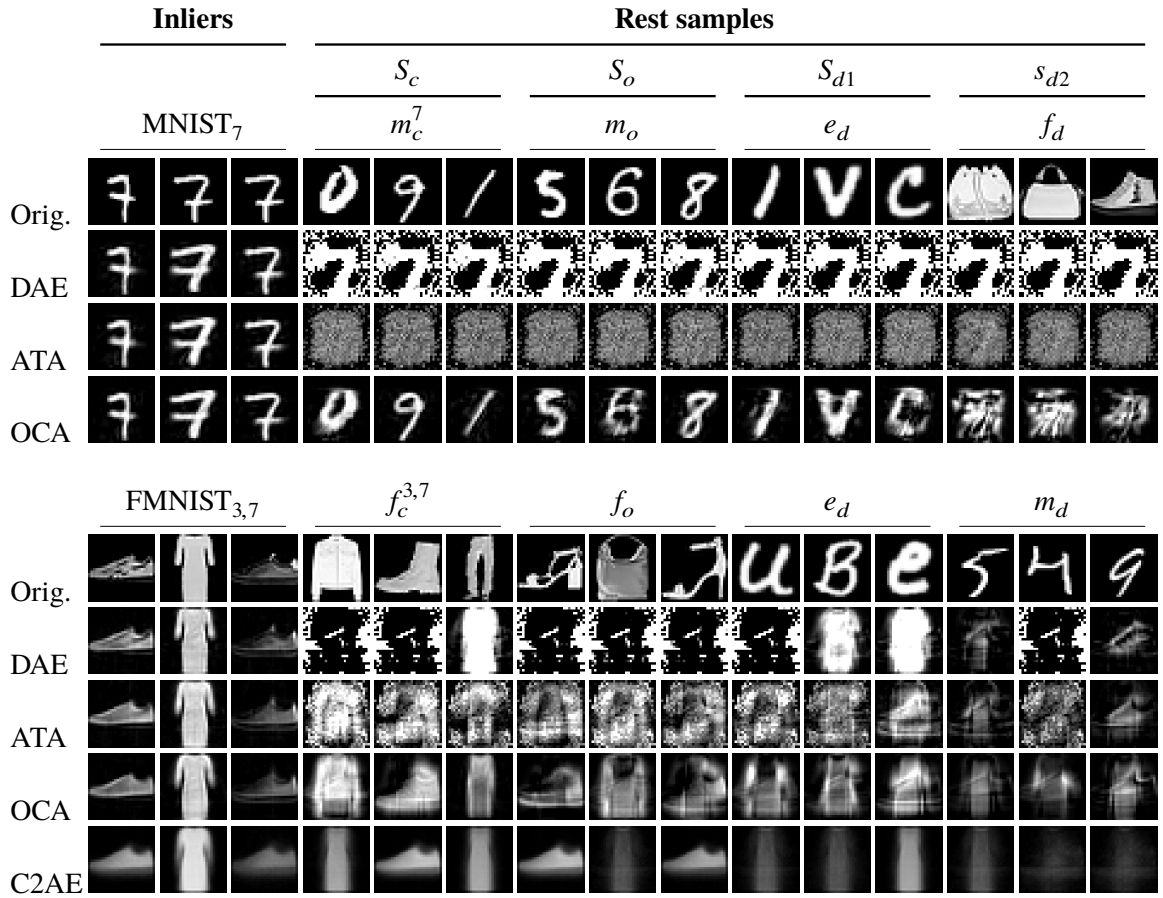


Figure 4.7: Qualitative reconstruction comparison of DAE and autoencoder-based baselines on $MNIST_7$ and $FMNIST_{3,7}$: The first row in each of the two grids, shows the original samples. We present three representative inlier samples, followed by three rest samples of the respective splits. In contrast to OCA and C2AE, our DAE method and ATA, with their adversarial loss functions, are able to not only accurately reconstruct inliers, but also obfuscate the rest samples.

	ATIS			REUTERS			Newsgroups		
	CECE ↓	Brier ↓	ΔCE ↓	CECE ↓	Brier ↓	ΔCE ↓	CECE ↓	Brier ↓	ΔCE ↓
DAE	3.1 ± 2.2	3.2 ± 1.9	24.8	1.2 ± 0.2	1.3 ± 0.2	23.7	4.6 ± 0.3	4.5 ± 0.3	11.3
MLP	2.8 ± 0.7	2.9 ± 0.7	28.6	1.3 ± 0.3	1.3 ± 0.3	28.5	4.8 ± 1.3	4.5 ± 0.9	15.3
MiMo	1.9 ± 0.6	2.9 ± 0.1	25.0	1.7 ± 0.8	1.7 ± 0.1	26.7	2.4 ± 1.3	3.4 ± 0.6	14.7

Table 4.5: Calibration of DAE, MLP and MiMo on split S_c in terms of CECE, Brier score and average per-bin calibration error difference ΔCE between T_c and T_o/T_d . All models are similarly well-calibrated on the classification task. The calibration similarity across splits is higher for DAE compared to the other methods. Metrics and confidence are measured in %.

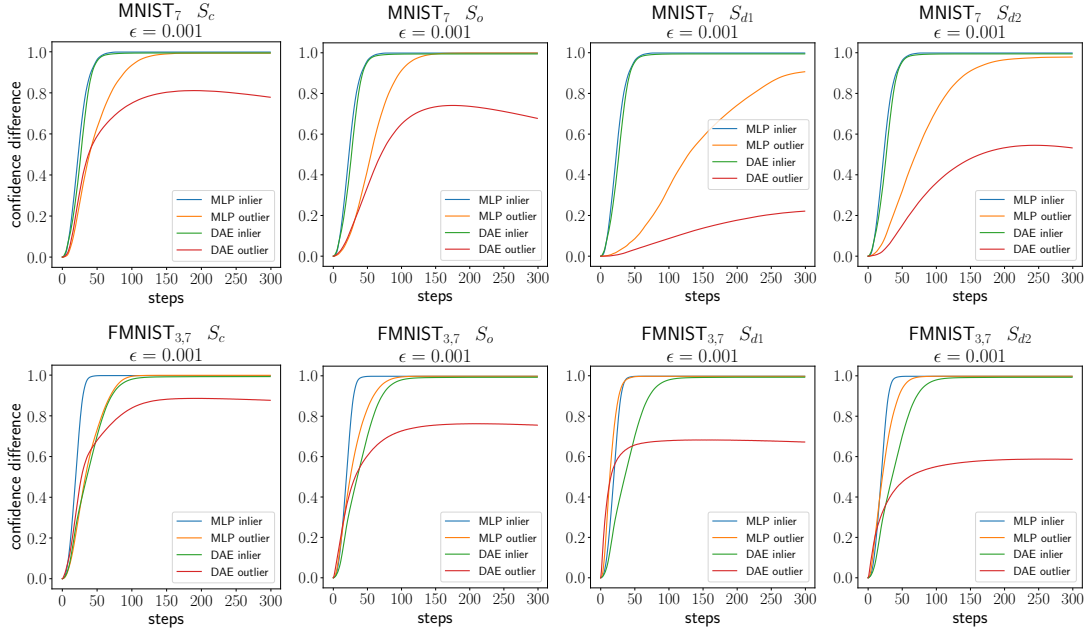


Figure 4.8: Adversarial robustness comparison between DAE and MLP: For each of the splits S_c , S_o , and S_d in the $MNIST_7$ and $FMNIST_{3,7}$ datasets, the samples (true positives and true negatives with more than 95% confidence) were perturbed via FGSM in 300 steps with a step size of 0.001, resolving the full range from local stability to global adversarial robustness. The results clearly show that DAE is more robust to adversarial perturbations applied to inliers and, especially, rest samples. On the two datasets, DAE is most robust w.r.t. adversarial perturbations on dataset shift samples, demonstrating the advantage of DAE being a true OSR method with bounded open space risk.

of calibration between T_c and T_o/T_d , since high similarity suggests that calibration improvements on T_c could generalize better to T_o and T_d . Technically, we measured the calibration error within each bin $\text{bin-CE}_{i,j}$ (see Eq. (4.19)), and then calculated the average per-bin calibration error difference ΔCE between split S_c and the S_o/S_d splits. Across all three datasets, DAE has a significantly lower ΔCE compared to MLP and MiMo. This is an interesting insight, as it suggests that for DAE, calibration deficiencies on T_c are more similar to deficiencies on T_o/T_d , compared to the other approaches and calibration improvement on T_c could generalize to calibration improvements on T_o and T_d . Multiple postprocessing-based approaches for calibration such as histogram binning [212] and temperature scaling [77] have also been proposed which could exploit the calibration similarity across tasks.

As shown in Table 4.6, we performed an ablation study w.r.t. the different loss terms in \hat{L} to show that only the specific combination in \hat{L} leads to the desired classification and robustness properties. The results clearly show that the loss function \hat{L} has the highest robustness with a minor classification degradation. If we remove the classification loss term L_{BCE} from \hat{L} , then the decision boundary t converges to 0, which classifies all samples as outliers, irrespective of their true class. If the model is solely trained on L_{BCE} or $L_{\text{BCE}} + |t|$, then the classification performance increases slightly, however, this still expresses severe robustness deficiencies to outliers and dataset shift.

These findings can be explained by Fig. 4.9, which plots the loss histograms for each hyperparameter combination in Table 4.6. The strong robustness performance of \hat{L} can be attributed to the wide

Hyperparameters					Figures	S_c		S_o		S_{d1}		S_{d2}	
λ_0	λ_1	λ_2	λ_3	Interpretation		AUROC	F1 score	AUROC	F1 score	AUROC	F1 score	AUROC	F1 score
1	0.01	0.01	1	\hat{L}	Figures 4.9(a) to 4.9(d)	98.9	97.5	98.9	96.9	99.0	98.4	99.0	97.2
1	0.01	0	1	$L_R + t $	Figures 4.9(e) to 4.9(h)	0.5	0	0.5	0	0.5	0	0.5	0
1	0	0	1	inlier and $ t $ minimization	Figures 4.9(i) to 4.9(l)	0.5	0	0.5	0	0.5	0	0.5	0
0	0	1	0	L_{BCE}	Figures 4.9(m) to 4.9(p)	99.2	97.1	98.2	91.7	71.0	66.9	91.7	76.3
0	0	1	1	$L_{BCE} + t $	Figures 4.9(q) to 4.9(t)	99.7	97.9	98.3	92.5	89.1	77.9	96.9	85.0

Table 4.6: Ablation study on MNIST_{2,7} w.r.t. the loss terms in \hat{L} controlled by hyperparameters λ_i (see Eq. (4.2) and Eq. (4.5)): The respective loss histograms are shown in Fig. 4.9. These results clearly indicate that the combination of all loss terms yields the highest dataset shift robustness, with slight degradation in classification performance. Without the adversarial loss term (i.e., $\lambda_1 = 0$), the models express significant robustness deficiencies and the two cases without L_{BCE} lead to unusable results, as t becomes 0. Note that the F1 scores can deviate from previous results in Table 4.4, as there the best models were selected based on AUROC scores on S_c . If the F1 score is a concern, we suggest filtering models whose decision boundary t has converged to a constant value, and subsequently select the best model based on AUROC.

inlier/outlier separation and small decision boundary t , which allows the model to reject outliers effectively, as shown in Fig. 4.9(a) to Fig. 4.9(d). Interestingly, L_{BCE} (Fig. 4.9(m) to Fig. 4.9(p)) and $L_{BCE} + |t|$ (Fig. 4.9(q) to Fig. 4.9(t)) can separate inliers from rest classes on S_c , but fail to generalize to unseen classes. In particular, without $|t|$ regularization, the inlier reconstruction errors are less minimized, leading to dataset shift samples becoming indistinctive from inliers (see Fig. 4.9(o)). If L_{BCE} is jointly optimized with $|t|$ regularization, then the minimization of inliers improves, but outliers are maximized less due to the vanishing gradient problem, as derived in Eq. (4.8). This results in poor OOD data robustness (see Fig. 4.9(s)). The adversarial component within L_{R2} does not suffer from the vanishing gradient limitation, and enforces the maximization of outliers, which becomes apparent when comparing Fig. 4.9(a) to Fig. 4.9(q). If the adversarial component is missing in L_R , then only inliers are minimized, which causes a significant overlap of inliers and rest classes, especially on S_c (see Fig. 4.9(i)). In conclusion, the combination of all loss terms in \hat{L} yields the best separation of inliers and outliers, due to effective minimization of inliers and maximization of outliers, as well as solving the vanishing gradient problem. Moreover, the minimization of the decision boundary t with L_{BCE} acting as an antipole, enables the model to robustly reject outliers without jeopardizing classification performance.

From this extensive analysis, we can conclude that DAE, as an OSR method with a bounded open space risk, clearly shows its superiority compared to the potent baselines from OSR, OVR, and outlier detection. Apart from ATA, every baseline consistently failed on more than one of the three subtasks of OSR, questioning their applicability in safety-critical systems. The consistent classification performance across all three tasks T_c , T_o and T_d combined with an increased (adversarial) robustness, shows the benefits of DAE’s reduced and bounded open space risk, and exposes the deficiencies of the ERM and semi-supervised baselines.

4.8 Conclusion

Open set recognition (OSR) is a common task in machine learning applications. Whenever the objective is to distinguish at least one class of interest (COI) from all remaining, possibly unknown classes (RC), e.g., ordinary internet traffic vs novel intrusion attempts, or general discussions vs all types of hate speech, OSR methods seem natural. Our analysis reveals that, when extending the

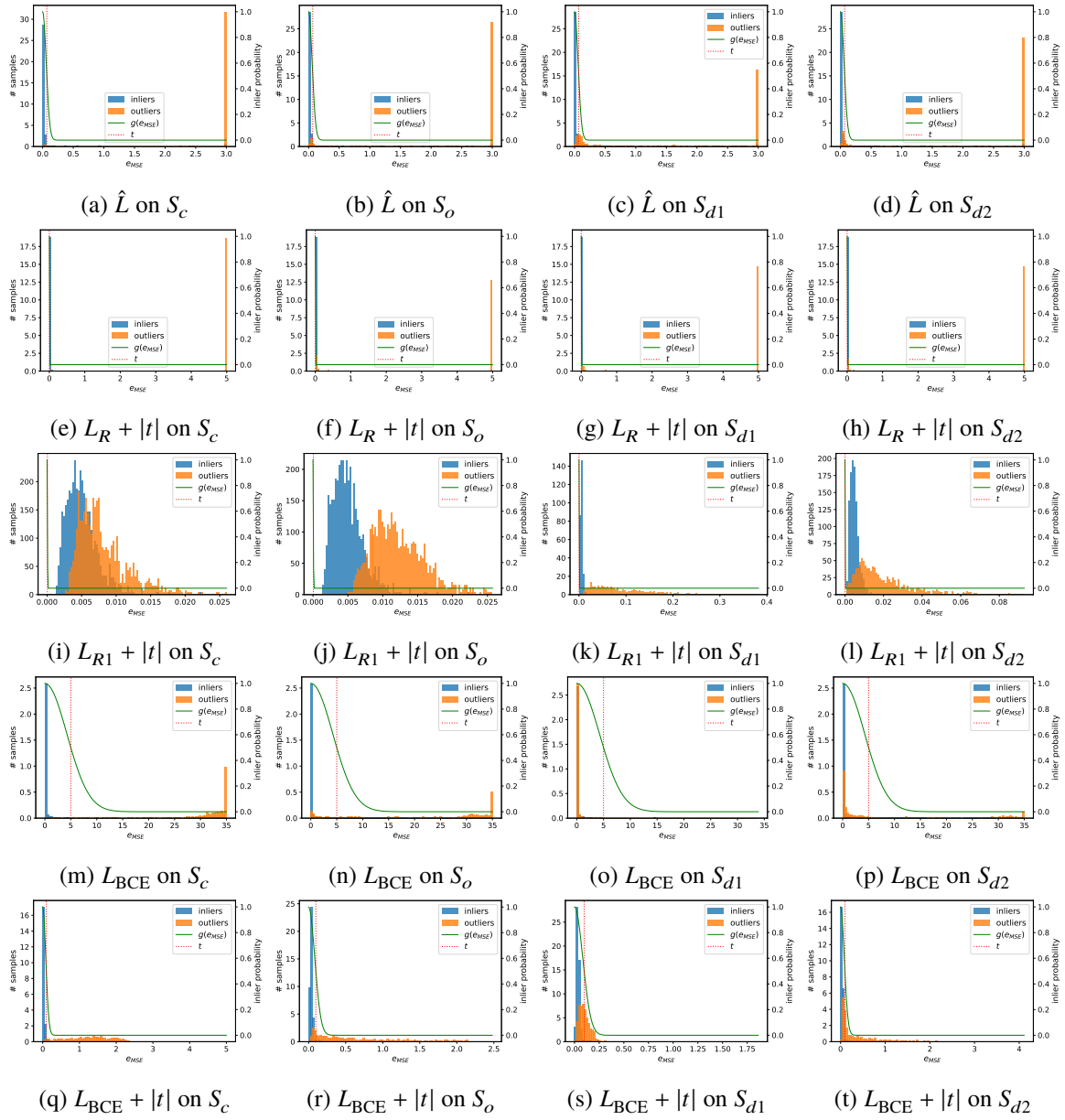


Figure 4.9: Inlier and outlier loss histograms regarding the different cases within the ablation study in Table 4.6. The model output $g(e_{MSE})$, which maps the reconstruction errors to inlier probabilities is plotted in green, and the decision boundary t is plotted as a dotted red line. Note that in some cases, e.g., in Fig. 4.9(a) to 4.9(d), the maximum reconstruction errors are capped to highlight the accuracy of the decision boundary. The results show that the combination of loss terms \hat{L} leads to inliers and outliers being minimized/maximized, respectively, while the decision boundary being as close to the inliers as possible, maximizes robustness without jeopardizing classification performance. All the other combinations lead to poor classification performance or lack of outlier robustness. Without outlier maximization, similar to the *one-class autoencoder*, the inliers and outliers are not fully separated on S_c and S_o (Fig. 4.9(i) to 4.9(l)).

scope of RC, OSR poses the difficult challenges of outlier detection and dataset shift to deep neural networks, solely optimized via empirical risk minimization. We provide an effective solution to these deficiencies with our proposed *decoupling autoencoder* (DAE) architecture. We have proven the existence of a bounded open space risk for DAE, and shown its classification and (adversarial) robustness benefits across three different subtasks of OSR. Specifically, we benchmarked DAE against capable baselines from various domains (DNNs, ensemble methods, outlier detection, and OSR) w.r.t. the OSR subtasks of classification, outlier detection, and dataset shift. In these experiments, DAE showed superior robustness across all subtasks, compared to the baselines, which failed on at least one of the tasks, apart from ATA. In comparison to ATA, DAE can actively minimize the open space risk, and does not require an offline brute-force line search for decision boundary estimation.

For future work, we would like to extend DAE towards multi-class classification, with a bounded open space risk, which would allow for robust multi-class classification under extreme dataset shift conditions. Another promising idea is the development of feature extractors that prevent the model from learning representations of noisy or uninformative features, thereby further alleviating the trade-off between classification and robustness performance.

Having fulfilled the second milestone defined in Sec. 1.5, we turn towards robust multi-class classification, seeking to provide a solution to the third and final milestone. Here, the main idea is to combine multiple DAEs in an ensemble, where each DAE solves one of the one-vs-rest classification problems, thereby also solving the multi-class classification problem, while maintaining the robustness benefits developed in this chapter.

From Open Set Recognition Towards Robust Multi-class Classification

The challenges and risks of deploying deep neural networks (DNNs) in the open-world are often overlooked and can potentially result in severe, negative outcomes, as explored in the previous chapters. Rather than improving the robustness to out-of-distribution data on a hit-or-miss basis, in this chapter we explore subjective uncertainty estimation of DNNs, as an important device for the interpretability of predictions. In case of a model being uncertain about its prediction, there are different actions, such as human supervision or increase of data coverage, that can provide the necessary information for the final prediction/decision. However, this kind of feedback loop can be only established for well-calibrated models. As pointed out in Sec. 1.1, traditional DNN methods are incapable of measuring uncertainty related to unforeseen circumstances accurately, providing wrong predictions at an elevated confidence level.

Technically, the central idea here is to leverage the DAE architecture engineered in Ch. 4 with its high robustness gains, and propose the *Informer* architecture, an extension towards multi-class classification. Architecturally, we create an ensemble from multiple DAEs, each learning a different one-vs-rest setting, and introduce a novel uncertainty estimation module that captures different sources of uncertainties.

Besides maintaining DAE's robustness properties, the new architecture captures the two orthogonal sources of uncertainty, namely epistemic and aleatoric uncertainty. While epistemic uncertainty can be reduced by the targeted sampling of data points the model has not been exposed to, i.e., unknown subspaces in the input space, aleatoric uncertainty cannot be reduced as it captures the statistical uncertainty within the data itself (e.g., uncertainty about rolling of fair dice cannot be reduced with an increased number of experiments). Therefore, considering these two uncertainty sources independently within the model predictions, allows for targeted retraining, enhancing a model's classification performance and robustness when few training samples are present. This also allows different actions to be taken, based on the separated uncertainty information within the prediction during deployment.

We evaluate the *Informer* architecture over a range of classification, outlier exposure and dataset shift exposure tasks, benchmarking *Informer* against potent baselines based on DNN ensembles, kernel-based DNNs, and traditional MLPs. Our results clearly show *Informer*'s superiority compared to these baselines in terms of robustness to outliers and dataset shift while maintaining a competitive

classification performance. Finally, we also empirically demonstrate that Informer can estimate the overall uncertainty within a prediction, and break the uncertainty estimate down into aleatoric and epistemic uncertainty in contrast to any of the other baselines. This is an essential feature in many use cases, as the underlying reasons for the uncertainty are fundamentally different, and can require different actions.

With the Informer architecture, we provide an effective solution to the last milestone, specified in Sec. 1.5.

This chapter is based on our publication [26]. The idea to combine multiple DAEs towards the Informer ensemble architecture for robust multi-class classification was proposed and implemented by Max Lübbering. Further, its ability to separate aleatoric and epistemic uncertainty was recognized and derived by Max Lübbering, who also conducted all experiments that resulted in the insights mentioned in the publication. The implementation of the MiMo [174] and DUQ [204] baseline were shared by Max Lübbering and one co-author. The Deep Ensembles [35] baseline was implemented by Max Lübbering. The paper was written by Max Lübbering and revised by the remaining co-authors, who also participated in the regular discussions concerning the Informer architecture.

5.1 Introduction

As pointed out in the previous chapters, the robustness of multi-class classifiers is an increasingly crucial, yet, underexplored criterion. The tremendous progress and outstanding performance of deep learning methods has led to their application in many parts of our daily lives, e.g., autonomous driving [213], and medical diagnosis [214]. These safety-critical applications have raised various concerns about AI safety [18, 19], as discussed in Sec. 1.1. Firstly, these concerns can be attributed to controlled benchmark environments that often do not reflect the deployment scenarios in an open-world setting, neglecting omnipresent side effects such as noise, dataset shift, or outlier exposure. Secondly, the empirical risk minimization (ERM) in the training process of DNNs has historically solely focused on the separation of the observed classes, leading to wrong and overly confident predictions on out-of-distribution (OOD) data [33, 35, 77, 215]. This is visualized in Fig. 5.1 for the two ensemble methods Deep Ensemble (DE) [35] and MIMO [174], and a traditional MLP with Softmax output. Many incidents that can be attributed to these shortcomings have been reported in the AI Incident Database [21] and analyzed in Ch. 1, stressing the importance of robust DNN methods.

Different concepts have been proposed to formalize the transition from closed-set classification to open-world classification. Most prominently, the open set recognition (OSR) framework, introduced in Ch. 4, aims to distinguish a set of inlier classes (observed closed set) from all rest classes (partially observed classes that are contextually related to the inlier classes and unobserved OOD classes)[25]. Here, the optimization objective is formalized as the two-fold problem of robustly rejecting OOD data while maintaining accurate classification between inliers and observed rest samples. Technically, the OSR framework jointly minimizes the empirical risk and open space risk [25, 179], which has led to different DNN architectures that robustly perform this generalized one-vs-rest (OVR) classification task [22, 58].

Another common approach towards robust DNNs, is to leverage their subjective predictive uncertainty. Conceptually, predictive uncertainty can be divided into aleatoric and epistemic uncertainty [27, 216]. Aleatoric uncertainty is referred to as the irreducible part of uncertainty, which is induced by statistical uncertainty within the data. A classic example is the uncertainty of rolling

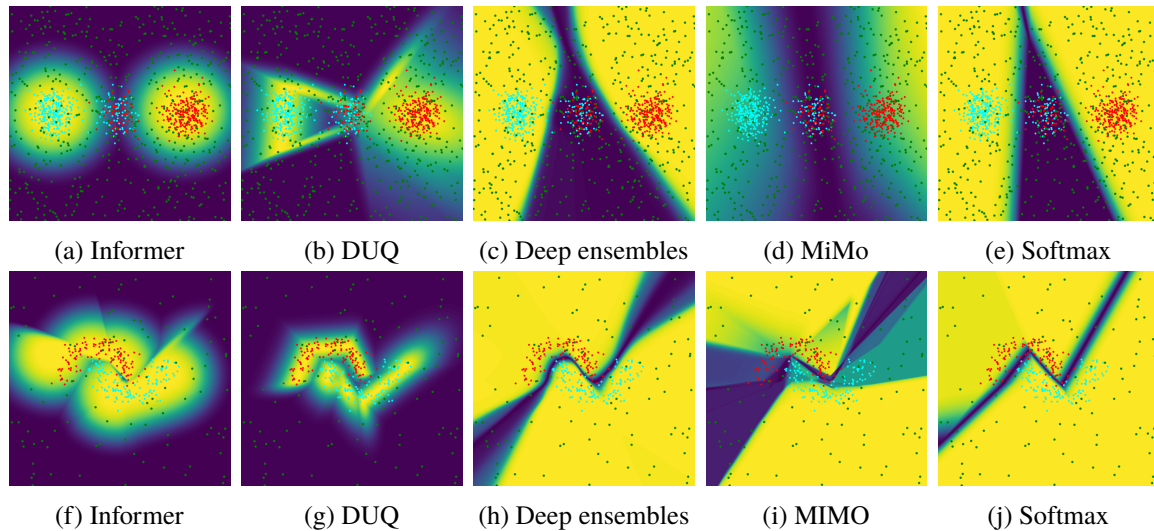


Figure 5.1: Uncertainty heatmap of our Informer architecture and the baselines on the half-moon dataset. Informer and DUQ capture aleatoric and epistemic uncertainty whereas the other three ERM baselines only capture aleatoric uncertainty.

fair dice that is irreducible despite an arbitrary long history of observations. Epistemic uncertainty quantifies the uncertainty induced by lack of knowledge about the best model for the given problem. Therefore, epistemic uncertainty is generally high on OOD data and, in contrast to aleatoric uncertainty, can be reduced with an increase in training data. Capturing and distinguishing these two types of uncertainties is crucial in safety-critical environments such as autonomous driving. In case of high epistemic uncertainty, the system would query more data from other sensors for uncertainty reduction [27], or if more data is unavailable, apply emergency braking, as the model is operating blindly. In the case of high aleatoric uncertainty, the model can make an informed decision based on risk assessment. Traditional DNNs trained with ERM are incapable of capturing epistemic uncertainty [27, 216], exposing humans and other entities engaging with such a system to an unforeseeable risk.

To this end, we propose *Informer*, a DNN architecture based on the OSR method *decoupling autoencoder* (DAE), introduced in Ch. 4. DAEs are trained in an adversarial fashion, which minimizes/maximizes the reconstruction error for inliers/outliers by applying gradient descent and gradient ascent, respectively. Thus, the reconstruction error becomes predictive of the inlierness of a sample. We extend this approach to multi-class classification by ensembling one DAE for each class and training them end-to-end by introducing a custom loss function that jointly optimizes classification and outlier robustness performance. Furthermore, we propose a practical uncertainty estimation module (UEM) that captures the overall uncertainty within a prediction and distinguishes aleatoric and epistemic uncertainty. Thus, this method provides interpretable uncertainty estimates; hence its name *Informer*.

In this chapter, we benchmark our method against seven potent DNN baselines, including the two ensembles MIMO [174] and DE [35], kernel method DUQ [204] and a feed forward Softmax model, on four image and text classification datasets. The key differences between these methods can be seen in Fig. 5.1, on the half-moon dataset. There, the ensemble methods and Softmax separate the inlier classes, but neglect uncertainties concerning outliers. In contrast to these, Informer and DUQ

learn a hull around the inlier classes, allowing them to reject outliers. Throughout our experiments, Informer expresses the highest robustness to corruptions, while maintaining a competitive classification performance. Finally, we empirically show that the UEM accurately distinguishes aleatoric and epistemic uncertainty.

5.2 Related Work

Enabling DNNs to capture and distinguish different uncertainty types is subject to an ongoing research, with multiple methods proposed from different domains [27]. These DNN methods can be categorized into Bayesian neural networks (BNNs), ensemble methods, and kernel methods. Most prominently, BNNs, with their long-lasting history of uncertainty estimation [216, 217], have been extended to quantify aleatoric and epistemic uncertainty [215, 218, 219]. Their reliance on variational inference, however, limits these methods to small-scale solutions.

Furthermore, [35, 174] have shown that simple, deterministic ensemble methods outperform BNNs, but their downsides are that these methods only provide an overall uncertainty score, which mostly captures aleatoric uncertainty and lacks principled retrieval of epistemic uncertainty.

As a solution, deep learning methods based on distance-aware kernels have been proposed [204, 220, 221]. The work of [221] combines spectral normalization for bi-Lipschitz regularization of residual blocks with a Gaussian process output layer to increase the sensitivity for epistemic uncertainty. Similarly, DUQ [204] regularizes a two-sided Lipschitz constraint within its loss function that is more relaxed than spectral normalization [220].

Various methods for OOD robustness have been proposed based on DNN outlier detectors [22, 23] and OSR methods [58], but only a few attempts leverage their sensitivity to epistemic uncertainty to improve DNN uncertainty estimation [27]. This is a gap we would like to bridge with our contribution, in this chapter.

5.3 Informer

To achieve robust, multi-class classification with accurate, predictive aleatoric and epistemic uncertainty estimates, we propose the Informer architecture based on a composition of DAEs [22], hereafter referred to as Informer components (ICs). Given a multi-class classification problem of classes $C = \{c_1, c_2, \dots, c_k\}$, the architecture comprises k ICs, each learning a different one-vs-rest (OVR) relationship for one of the classes. As shown in Fig. 5.2, the IC of inlier class c_i is composed of encoder e_i and decoder d_i , which reconstruct sample $\mathbf{x} \in \mathbb{R}^n$ as reconstruction $\hat{\mathbf{x}}^{(i)} \in \mathbb{R}^n$. The non-parameterized reconstruction error module e_{MSE} maps a sample/reconstruction pair onto a scalar reconstruction error value, which is subsequently mapped to probability $p(c_i|\mathbf{x}; \Theta^{(i)}, \sigma_i)$ via the Gaussian kernel g parameterized with standard deviation σ_i . Thus, the subnetwork IC_i is given by

$$p(c_i|\mathbf{x}) = g(e_{\text{MSE}}(d_i(e_i(\mathbf{x}; \Theta_e^{(i)}); \Theta_d^{(i)}), \mathbf{x}), \sigma_i), \quad (5.1)$$

where Gaussian $g(z) = e^{-\frac{z^2}{2\sigma_i^2}}$, the error module $e_{\text{MSE}}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{n} \sum_j^n (x_j - \hat{x}_j)^2$ and $\Theta_e^{(i)}, \Theta_d^{(i)}$ denote the network weights of the encoder and decoder part of IC i , respectively. Note that for readability purposes, we omit the explicit naming of parameters Θ and σ in the subsequent formulas when it

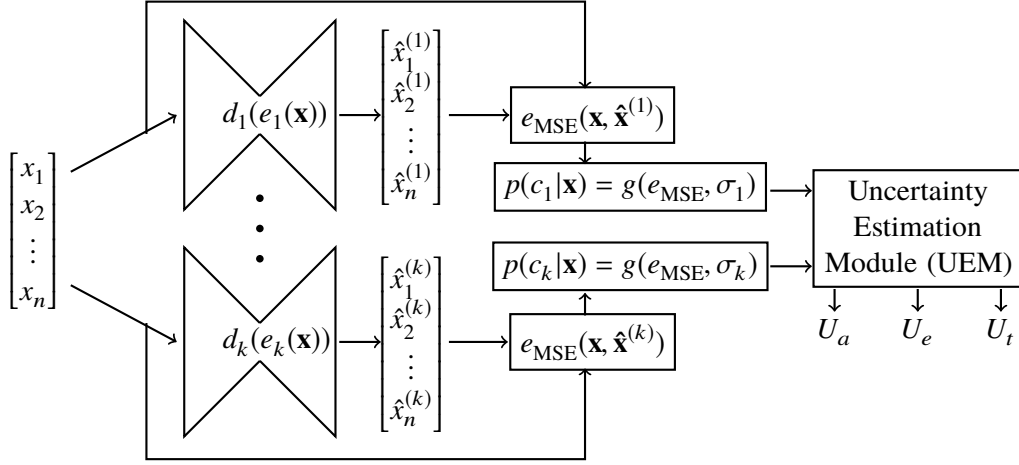


Figure 5.2: Illustration of the Informer architecture for multi-class classification with k classes: An ensemble composed of k Informer components (autoencoder $d_i(e_i(\mathbf{x}))$ for sample reconstruction, a reconstruction error module e_{MSE} for outlieriness estimation, and an RBF kernel g for classification), each learning an OVR relationship for a different inlier class. The uncertainty estimation module (UEM), provides estimates on the total, aleatoric and epistemic uncertainty, as denoted by U_t , U_a and U_e , respectively.

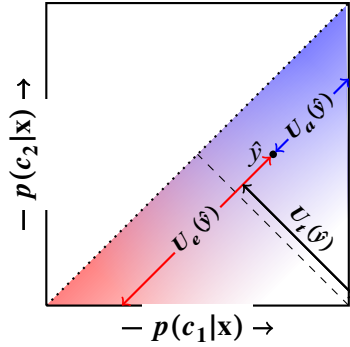


Figure 5.3: Uncertainty Estimation Module (UEM): Given a prediction \hat{y} , the two class probabilities $p(c_1|\mathbf{x})$ and $p(c_2|\mathbf{x})$, conditioned on the input \mathbf{x} , determine the amount of aleatoric and epistemic uncertainty within the prediction. UEM splits the uncertainty U_t (Manhattan distance of prediction to bottom right corner) into a convex combination of aleatoric uncertainty U_a and epistemic uncertainty U_e . For simplicity, we only show the case $p(\hat{y}_1|\mathbf{x}) > p(\hat{y}_2|\mathbf{x})$.

does not compromise clarity. Based on the OVR probability $p(c_i|\mathbf{x})$ of each class c_i , UEM provides estimates on the total prediction uncertainty and the aleatoric/epistemic uncertainty ratio. The Informer network is defined as $\varphi(\mathbf{x}) = (p(c_1|\mathbf{x}), \dots, p(c_k|\mathbf{x}))^T$ where $\sum_i^k p(c_i|\mathbf{x}) = 1$ is not enforced.

To train the Informer architecture, we adapted the original DAE loss function, which is optimized for the open set recognition task, such that it learns the multi-class classification problem while maintaining its robustness benefits. The derived loss function \hat{L} is given by

$$\hat{L}(\mathbf{x}, \varphi(\mathbf{x}; \Theta), y) = \sum_i^k L_R(\mathbf{x}, d_i(e_i(\mathbf{x}; \Theta_e^{(i)}); \Theta_d^{(i)}), \mathbb{1}(y = i)) + \lambda_1 L_{\text{CE}}(\varphi(\mathbf{x}; \Theta), y) + \lambda_2 |\sigma|, \quad (5.2)$$

where $\lambda_1 \in \mathbb{R}$ and $\lambda_2 \in \mathbb{R}$ scale the cross-entropy loss L_{CE} and the regularization of $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_k)^T$, respectively. The loss term L_R denotes the adversarial reconstruction error, as defined by us in [22] and introduced in Eq. (5.2) in Sec. 4.3, which minimizes/maximizes the reconstruction error for inliers/rest samples, respectively. The target $y \in C$ is mapped to $\{0, 1\}$ using indicator function $\mathbb{1}(y = c_i)$, reflecting the OVR relationship within a single IC. Technically, the reconstruction of rest samples is maximized by the negation of the mean squared error loss L_{MSE} , which corresponds to gradient ascent [23]. The classification cross-entropy loss L_{CE} , with its primary purpose to optimize $\boldsymbol{\sigma} \in \mathbb{R}^k$ across the different ICs, is defined as

$$L_{\text{CE}}(\varphi(\mathbf{x}; \Theta), y) = -\log(\text{softmax}_i(\varphi(\mathbf{x}; \Theta))), \quad (5.3)$$

where target $y = c_i$ and $\text{softmax}_i(\varphi(\mathbf{x}; \Theta)) = \frac{e^{\varphi(\mathbf{x}; \Theta)_i}}{\sum_j^k e^{\varphi(\mathbf{x}; \Theta)_j}}$. Since $\sum_i^k \varphi(\mathbf{x}; \Theta)_i = 1$ is not enforced, we apply softmax to the model output $\varphi(\mathbf{x})$, which is appropriate to the multi-class classification setting. While theoretically each IC could be trained independently, the learned decision boundaries imposed by $\boldsymbol{\sigma}$ can be incompatible due to calibration inaccuracies, e.g., if one IC is overly confident, classification performance is harmed. Naturally, traditional ensembles do not suffer from this issue, as each component is trained on all classes. Within Informers, we alleviate this problem by L_{CE} , due to the IC interdependence within softmax. Given an Informer network $\varphi(\mathbf{x}, \Theta, \boldsymbol{\sigma})$ with autoencoder weights Θ , decision boundary parameter $\boldsymbol{\sigma}$, and sample \mathbf{x} with target $y = c_i$, the gradient w.r.t. σ_i and σ_j with $j \neq i$ computes to

$$\frac{\partial L_{\text{CE}}(\varphi(\mathbf{x}, \Theta, \boldsymbol{\sigma}), y)}{\partial \sigma_i} = -\frac{\sum_{l \in \{0, \dots, k\} \setminus \{i\}} e^{\varphi_l(\mathbf{x}, \Theta, \boldsymbol{\sigma})}}{\sum_l^{\{0, \dots, k\}} e^{\varphi_l(\mathbf{x}, \Theta, \boldsymbol{\sigma})}} \frac{\partial \varphi(\mathbf{x}, \Theta, \boldsymbol{\sigma})}{\partial \sigma_i} \quad (5.4)$$

$$= -(1 - p(c_i | \mathbf{x})) \frac{\partial \varphi(\mathbf{x}, \Theta, \boldsymbol{\sigma})}{\partial \sigma_i} \quad (5.5)$$

and

$$\frac{\partial L_{\text{CE}}(\varphi(\mathbf{x}, \Theta, \boldsymbol{\sigma}), y)}{\partial \sigma_j} = \frac{e^{\varphi_j(\mathbf{x}, \Theta, \boldsymbol{\sigma})}}{\sum_l^{\{0, \dots, k\}} e^{\varphi_l(\mathbf{x}, \Theta, \boldsymbol{\sigma})}} \frac{\partial \varphi(\mathbf{x}, \Theta, \boldsymbol{\sigma})}{\partial \sigma_j} \quad (5.6)$$

$$= p(c_j | \mathbf{x}) \frac{\partial \varphi(\mathbf{x}, \Theta, \boldsymbol{\sigma})}{\partial \sigma_j}, \quad (5.7)$$

respectively. A full derivation can be found in Appendix A.1.1. Eq. (5.4) and Eq. (5.6) display two important gradient properties of L_{CE} : 1) The gradients point in opposite directions, resulting in minimization / maximization of σ_i and σ_j within gradient descent, respectively. 2) The gradient scaling factors can be interpreted as probabilities $1 - p(c_i | \mathbf{x})$ and $p(c_j | \mathbf{x})$, as per the definition of the softmax function. Thus, L_{CE} models the interdependence of $\boldsymbol{\sigma}$ within the multi-class classification setting.

The loss terms L_{CE} and $|\boldsymbol{\sigma}|$ optimize $\boldsymbol{\sigma}$, whereas L_R and L_{CE} optimize autoencoder weights Θ . Importantly, even though we apply L_{CE} to all network weights, there is no information flow between ICs at inference time, since $p(c_i | \mathbf{x})$ is independent of the weights of the other ICs $p(c_j | \mathbf{x}) \quad \forall i \neq j$. Hence, the two ICs can predict their inlier class with maximum confidence, interpreted as maximum

aleatoric uncertainty within the overall prediction.

Following up on this, we formalize the uncertainty estimation as a two-step approach. Firstly, given a sample \mathbf{x} with predictions $\hat{y} = \varphi(\mathbf{x})$, we filter the two class predictions with the highest certainty. Secondly, as shown in Fig. 5.3, we derive the aleatoric and epistemic uncertainty ratios from their location on the probability plane. The model is aleatorically uncertain if both probabilities are high (top right triangle corner), epistemically uncertain if both probabilities are low (bottom left triangle corner), and certain if only one of the probabilities is high (bottom right triangle corner). A prediction has the highest total uncertainty U_t if it lies on the diagonal between maximum epistemic uncertainty and maximum aleatoric uncertainty. Hence, U_t is a convex combination of aleatoric and epistemic uncertainty. Similarly, all points located on a parallel to this diagonal have equal U_t , as induced by the Manhattan distance from the point of maximum certainty. Thus, omitting simple linear algebra, the uncertainty scores can be calculated from the two maximum scores \hat{y}_1 and \hat{y}_2 within prediction $\varphi(\mathbf{x})$ with $\hat{y}_1 > \hat{y}_2$ as follows

$$U_t(\hat{y}) = \frac{\sqrt{\frac{(1+\hat{y}_2-\hat{y}_1)^2}{2}}}{\sqrt{\frac{1}{2}}}, \quad U_a(\hat{y}) = \frac{\hat{y}_2\sqrt{2}}{\sqrt{2(1+\hat{y}_2-\hat{y}_1)}}, \quad U_e(\hat{y}) = 1 - U_a(\hat{y}). \quad (5.8)$$

A full derivation of the uncertainty estimators can be found in Appendix A.1.2.

The Informer architecture has multiple advantages. While traditional DNNs generally only capture aleatoric uncertainty due to the adoption of ERM [27], the proposed Informer method can capture and distinguish aleatoric and epistemic uncertainty due to Θ independence between ICs and global σ optimization. This property is a crucial requirement when deploying models within the open world that is not met by the vast majority of DNN methods.

5.4 Evaluation Approach

The models are evaluated in three different scenarios (i.e., classification, contextual outlier exposure, and dataset shift exposure), as proposed by [22, 99] and applied in Ch. 4 for model evaluation within the OSR domain. These scenarios are represented by the test splits S_c , S_o , and S_d , respectively, as depicted in Table 5.1. We leverage the three image classification datasets FMNIST, MNIST, and EMNIST, and the three text classification datasets Reuters, Newsgroups and ATIS. The models are trained on FMNIST, MNIST, Reuters, and Newsgroups with the respective inlier classes $f_c = \{\text{t-shirt, pants, pullover, dress, sneaker}\}$, $m_c = \{0, 2, 4, 6, 8\}$, $r_c = \{\text{acq, earn, crude, interest, money-fx}\}$ and $n_c = \{\text{sci.med, rec.autos, sci.space, misc.forsale, rec.sport.hockey}\}$. The remaining unobserved classes from the respective dataset are added as rest samples to split S_o for contextual outlier exposure, thereby increasing aleatoric and epistemic uncertainty. Finally, we leverage rest classes from unrelated datasets within splits S_{d1} and S_{d2} , which primarily increases epistemic uncertainty in the data. As part of the preprocessing, the image samples have been z-transformed, and the text samples were embedded as pooled 100-dimensional Glove embeddings [156].

We evaluate the closed set classification performance in terms of the macro F1 score on split S_c , and outlier robustness on S_o and S_d in terms of AUROC. Since we expect models to be uncertain about rest samples, this characteristic is captured by AUROC, which can be interpreted as the probability of

Split	FMNIST		MNIST		Reuters		Newsgroups	
	Inlier	Rest	Inlier	Rest	Inlier	Rest	Inlier	Rest
S_t	f_c	–	m_c	–	r_c	–	n_c	–
S_c	f_c	–	m_c	–	r_c	–	n_c	–
S_o	f_c	$f \setminus f_c$	m_c	$m \setminus m_c$	r_c	$r \setminus r_c$	n_c	$n \setminus n_c$
S_{d1}	f_c	m	m_c	f	r_c	n	n_c	r
S_{d2}	f_c	e	m_c	e	r_c	a	n_c	a

Table 5.1: Three step experiment setup concerning classification (S_c), contextual outlier exposure (S_o) and dataset shift exposure (S_{d1} and S_{d2}): The two sets of inlier/rest classes within a split are denoted the first letter of the original dataset. The training split S_t and test splits (S_c, S_o, S_d) share the same inlier classes for a given dataset. The contextual outlier split S_o and dataset shift splits S_d provide the rest classes from the same dataset and an unrelated dataset, respectively. Note that the sets e and a refer to the classes of EMNIST and ATIS.

a random rest sample being ranked higher than a random inlier sample [22, 33] (see Sec. 1.2.1 for a discussion).

We selected an MLP as a strong classification baseline, and the two ensemble methods MIMO [174] and deep ensembles (DE) [35], with an improved uncertainty estimation in comparison to the MLP [35]. These are compared using two different offline uncertainty estimation methods based on the softmax outputs, as proposed by [33], namely $\text{entropy}(\text{softmax}(\varphi(\mathbf{x})))$ and $\max(\text{softmax}(\varphi(\mathbf{x})))$. The different variants are denoted by a leading E and M, e.g., E-MLP for entropy-based uncertainty estimation within MLP. Finally, Informer is benchmarked against the kernel-based DNN method DUQ [204], which was proposed for robust uncertainty quantization.

For an algorithm-level comparison, we applied nested CV. Each IC has a hidden layer sizes of [50, 25, 12, 25, 50] for text classification and [256, 128, 256] for image classification. For a fair comparison, all baselines can have a maximum model complexity comparable to the Informer complexity. The ensemble size of MIMO and DE are fixed to 5, matching the number of ICs. All models are optimized with Adam [32], while the Informer decision boundary σ is optimized via stochastic gradient descent (SGD) at a higher learning rate, due to the low optimization complexity. All methods are optimized w.r.t. *learning rate* and *weight decay*. Additionally, DUQ is optimized w.r.t. *gradient penalty* and *length scale*. Regarding Informer, we perform a sweep over scaling factors λ_i and the inlier/outlier weighting factor within loss term L_R .

The best Informer model is selected in two steps: 1) The model configurations whose decision boundary has converged to a static value are selected. 2) The final model is chosen by the highest macro F1 score. While there are many possible decision boundaries in reconstruction error space that accurately split the classes, we argue that the smallest decision boundary leads to the tightest hull around the inlier classes, and the best generalization towards outlier robustness. For DUQ, we select the best model by the involved, multi-step approach proposed by its authors [204] and for the remaining baselines, we select the best one via macro F1 score.

Method	FMNIST				MNIST			
	F1 score		AUROC		F1 score		AUROC	
	S_c	S_o	S_{d1}	S_{d2}	S_c	S_o	S_{d1}	S_{d2}
Informer	96.9 ± 0.3	78.4 ± 0.6	95.8 ± 1.0	95.7 ± 1.1	99.0 ± 0.1	87.9 ± 2.4	99.5 ± 0.2	93.9 ± 0.8
DUQ	96.8 ± 0.3	72.4 ± 3.2	91.3 ± 2.3	90.9 ± 1.6	99.0 ± 0.1	91.4 ± 0.2	92.6 ± 1.1	91.2 ± 0.1
E-MLP	96.7 ± 0.2	52.1 ± 1.7	77.9 ± 1.8	76.7 ± 2.2	98.8 ± 0.1	85.1 ± 3.3	83.6 ± 4.9	82.2 ± 2.8
M-MLP	96.7 ± 0.2	52.6 ± 1.8	77.6 ± 1.8	76.4 ± 2.2	98.8 ± 0.1	85.1 ± 3.3	83.7 ± 4.9	82.2 ± 2.8
E-MIMO	95.6 ± 0.2	73.5 ± 1.4	95.7 ± 1.2	93.7 ± 1.1	96.7 ± 0.3	85.5 ± 1.4	92.1 ± 1.7	86.1 ± 2.3
M-MIMO	95.6 ± 0.2	73.2 ± 1.2	95.2 ± 1.1	93.4 ± 1.0	96.7 ± 0.3	85.6 ± 1.5	92.2 ± 1.6	86.1 ± 2.3
E-DE	96.9 ± 0.2	55.8 ± 1.4	91.2 ± 1.3	88.0 ± 2.1	99.1 ± 0.1	89.7 ± 1.6	86.9 ± 2.9	88.7 ± 1.1
M-DE	96.9 ± 0.2	55.9 ± 1.6	91.0 ± 1.2	87.8 ± 2.0	99.1 ± 0.1	89.7 ± 1.6	86.9 ± 2.8	88.6 ± 1.1

Table 5.2: Image classification results on the different test splits: The best performing score is highlighted in boldface for each split. Weak performances are highlighted in gray within each split when the score deviates more than 10 percentage points from the best score. The Informer architecture provides competitive classification results while being most robust to outliers/dataset shift.

Method	Reuters				Newsgroups			
	F1 Score		AUROC		F1 Score		AUROC	
	S_c	S_o	S_{d1}	S_{d2}	S_c	S_o	S_{d1}	S_{d2}
Informer	90.0 ± 1.6	83.7 ± 2.2	93.7 ± 0.9	91.0 ± 2.0	91.9 ± 0.7	78.8 ± 1.2	88.1 ± 4.7	87.0 ± 5.9
DUQ	89.2 ± 2.5	80.8 ± 5.3	88.2 ± 3.2	95.1 ± 1.0	86.7 ± 5.6	73.6 ± 4.5	70.6 ± 8.0	68.8 ± 8.0
E-MLP	89.5 ± 1.6	63.0 ± 8.3	62.6 ± 8.5	83.2 ± 4.5	91.6 ± 0.8	82.1 ± 1.2	53.7 ± 8.2	68.2 ± 3.1
M-MLP	89.5 ± 1.6	62.4 ± 8.3	62.2 ± 8.4	82.4 ± 4.6	91.6 ± 0.8	81.9 ± 1.2	54.4 ± 8.2	68.9 ± 3.4
E-MIMO	89.0 ± 1.3	87.7 ± 2.8	92.5 ± 1.6	97.3 ± 1.1	89.3 ± 0.9	80.1 ± 1.2	76.4 ± 4.0	77.9 ± 3.7
M-MIMO	89.0 ± 1.3	87.0 ± 2.9	91.8 ± 1.7	96.2 ± 1.4	89.3 ± 0.9	80.2 ± 1.2	78.0 ± 3.8	79.2 ± 3.9
E-DE	90.1 ± 2.0	80.1 ± 6.1	86.6 ± 3.4	96.2 ± 0.3	91.7 ± 1.0	82.6 ± 0.7	74.7 ± 1.9	75.4 ± 2.5
M-DE	90.1 ± 2.0	79.3 ± 5.9	86.0 ± 3.4	95.1 ± 0.4	91.7 ± 1.0	82.2 ± 0.7	75.6 ± 1.9	76.1 ± 2.6

Table 5.3: Results on text datasets with the same score highlighting as in Table 5.2: Similar to the results on image datasets, Informer provides competitive classification without the robustness deficiencies of the other baselines.

5.5 Results

Having applied the aforementioned experiment setup, the results in Table 5.2 and Table 5.3 show that each method expresses similar performance on the classification split S_c . When exposed to contextual outliers and dataset shift, the MLP baseline expresses significant robustness deficiencies with a weak performance in 20/24 cases on S_o , S_{d1} and S_{d2} . MIMO, DE, and DUQ significantly improve the model robustness, but nevertheless, the Informer robustness scores exceed the performance of all baselines. Not only does Informer achieve the most top robustness scores, it also never yields any weak robustness scores. In conclusion, while all methods yield a similar classification performance, the subjective uncertainty scores of the Informer architecture are more sensitive to outliers, making

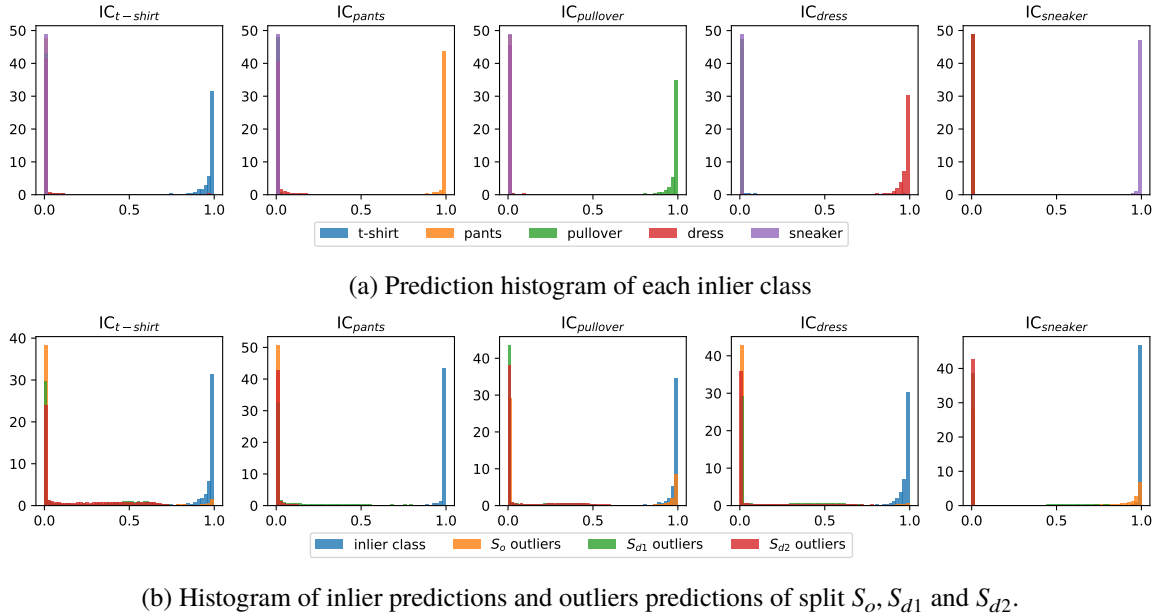


Figure 5.4: Histogram of IC predictions $\varphi_i(\mathbf{x}) = p(c_i|\mathbf{x})$: While the inlier class is generally well-separated from rest classes, there are few outlier samples primarily from split S_o that are falsely predicted as inliers with high confidence.

Informer the most robust method.

We further explored the robustness of each IC by comparing the histograms for the inlier classes and rest classes from the different splits. As shown in Fig. 5.4(a), each IC learns the OVR relationship, explaining the competitive classification results. As shown in Fig. 5.4(b), the inlier class is less separated from the S_o rest classes than the S_d rest classes. This suggests that contextual outliers are reconstructed more accurately than dataset shift samples, possibly allowing the method to capture aleatoric and epistemic uncertainty separately.

This finding is supported by Fig. 5.5(a), which visualizes samples of high aleatoric uncertainty in the top right corner and samples of high epistemic uncertainty in the bottom left corner, per Eq. (5.8). Some of the S_c and S_o samples express high aleatoric uncertainty, whereas S_d samples are predicted with high epistemic uncertainty and never yield high aleatoric uncertainty. The predicted epistemic and aleatoric uncertainty is well-aligned with human perception, as shown in Table 5.4. The two S_c samples with max U_a have middle-sized sleeves adding significant classification ambiguity even for the human eye. Interestingly, the S_c samples with max U_e cannot be assigned to any of the two classes, as they are mislabeled dresses, and thus correctly rejected by the model. Similar conclusions can be drawn from the selected S_o samples. S_{d1} samples with max U_a have comparably low aleatoric uncertainty and high U_t , further showcasing the method’s effectiveness of rejecting OOD samples.

In contrast to Informer, DUQ can only capture U_t and is unable to differentiate between different uncertainty types, as shown in Fig. 5.5(b), with samples of all splits indistinctly spread over the bottom left triangle. We presume that DUQ does not separate aleatoric and epistemic uncertainty, since the transformation from input to embedding space can produce artifacts that map OOD data to areas of higher aleatoric uncertainty in embeddings space. This deficiency is prevented within Informer by learning a tight hull around the inlier class within each IC, as shown by [22] for DAE. If these hulls

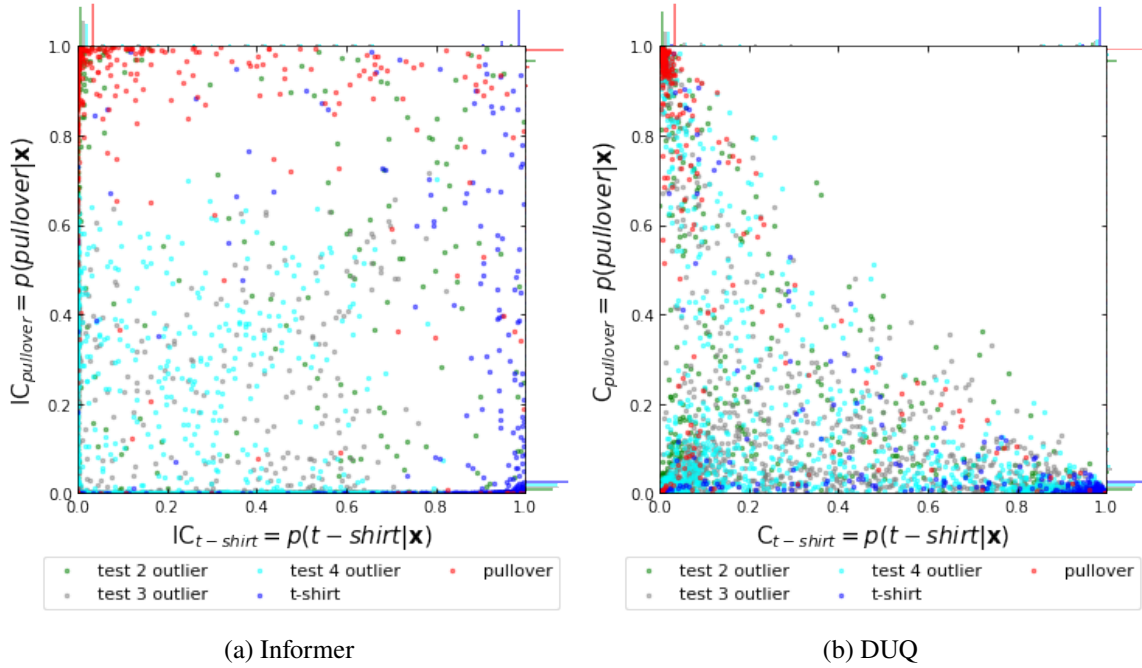


Figure 5.5: Comparing the prediction scatter of Informer (IC_{pullover} and $IC_{\text{t-shirt}}$) to DUQ’s centroids (C_{pullover} and $C_{\text{t-shirt}}$) w.r.t. inlier classes pullover/t-shirt and S_o/S_{di} rest classes. Due to the majority of samples overlapping in the dense corners, we added histogram bars on the top and right edge for visual support.

	S_e samples						S_o rest samples						S_{di} rest samples					
	min U_t		max U_e		max U_a		min U_t		max U_e		max U_a		min U_t		max U_e		max U_a	
Label	t-shirt	pull.	t-shirt	pull.	t-shirt	pull.	shirt	shirt	about	sandal	shirt	shirt	1	0	6	3	7	0
Orig.																		
IC 0																		
IC 1																		
$p(\text{t-shirt} \mathbf{x})$	100.0	0.0	0.0	0.0	95.8	96.1	99.9	0.7	0.0	0.0	99.4	98.3	92.8	0.0	0.0	0.0	65.7	67.6
$p(\text{pullover} \mathbf{x})$	0.0	99.9	0.0	0.0	96.4	95.5	0.0	99.9	0.0	0.0	98.9	99.0	0.0	71.9	0.0	0.0	51.8	49.1
U_t	0.0	0.0	1.0	1.0	0.99	0.99	0.0	0.01	1.0	1.0	1.0	0.99	0.07	0.28	1.0	1.0	0.86	0.82
U_a	nan	0.0	0.0	0.0	0.96	0.96	0.0	0.08	0.0	0.0	0.99	0.99	0.0	0.0	0.0	0.0	0.56	0.54
U_e	nan	1.0	1.0	1.0	0.04	0.04	1.0	0.92	1.0	1.0	0.01	0.01	1.0	1.0	1.0	1.0	0.44	0.46

Table 5.4: Representative samples and their reconstructions from each split, expressing either min U_t , max U_e or max U_a . The samples clearly show that different uncertainty types can be captured by the Informer architecture.

overlap, this is a strong indicator of true aleatoric uncertainty within the data, as seen in the examples in Table 5.4.

In conclusion, the Informer architecture yields competitive classification results over the full range of experiments, while being highly robust to outliers and dataset shift. This level of robustness is not observed for any of the other strong baselines. Furthermore, Informer can distinguish between aleatoric and epistemic uncertainty, which is impossible for the baselines. The subjective uncertainty predictions also match humans’ perceptions of ambiguity and unknownness. All of these insights

make this method compelling for use in open-world deployment scenarios.

5.6 Conclusion

Deploying models in the open world poses various challenges to traditional DNN methods. Generally, the empirical risk minimization is solely focused on inlier separation and does not consider dataset shift and outlier exposure prevalent in open-world scenarios, potentially leading to uncontrollable and harmful behavior in the open-world setting. To this end, we proposed the Informer architecture as an ensemble of autoencoder-based OVR classifiers that is highly robust to such corruptions. Furthermore, Informer can estimate the overall uncertainty within a prediction and subdivide the uncertainty into epistemic and aleatoric uncertainty in a principled way. Moreover, we demonstrated Informer’s robustness superiority over two deep ensemble methods [35, 174], the kernel-based uncertainty quantization method DUQ [204], and traditional MLPs, on multiple text datasets and image datasets, in various experiment settings. Finally, we empirically verified Informer’s capability of differentiating between aleatoric and epistemic uncertainty that is well-aligned with human perception. The Informer architecture provides an effective solution to the problem, defined as the last milestone in Sec. 1.5, concerning robust multi-class classification.

For future work, we consider adding a feature extractor to the front of the Informer architecture, that drops noisy/uninformative features, which would otherwise poise the inlierness signal of the autoencoders’ reconstruction error. Additionally, we would like to apply this approach to critical areas such as medical diagnosis, in which robustness and accurate uncertainty estimation are crucial.

Applications

In this chapter, we demonstrate the real-world applicability of our methods from an application and deployment point of view. Firstly, we develop a toxicity detection system for online communication, based on our proposed *adversarially trained autoencoder* (ATA) method, showcasing the benefits and limitations of our approach on real-world data. Secondly, we implement a document information extraction system illustrating how our algorithms can be leveraged for large-scale information extraction and integrated in real-world deployment scenarios.

More specifically, in the first part, we present a comparative study on toxicity detection, focusing on the problem of identifying toxicity types of low prevalence, some of which are unobserved at training time. This is an extremely difficult task for autoencoders, unmatched by the previous datasets, as toxicity is not restricted to a single topic and usually depends on the context. Given the manifold hypothesis introduced in Sec. 1.3, this requires the model to encode all the context information on a gigantic, latent manifold, to then accurately decode the latent representation. To further increase the complexity, we train ATA and the baselines only on a weak type of toxicity and non-toxic samples, and test whether ATA is able to generalize to more severe toxicity types. Our results suggest that ATA and the ensemble baseline exceed the classification performance of simple classifiers on toxicity detection, while also providing significantly better generalization and robustness. All models benefit from a larger training set size, which even extends to the toxicity types unseen during training.

The second part deals with a deployment case study regarding the use case of information extraction from financial documents. Generally, the problem of extracting information from large volumes of unstructured documents is pervasive in the financial domain. Enterprises and investors rely on automatic methods that can extract information from financial documents, particularly for indexing and efficiently retrieving information. To achieve this level of automation, we present a scalable and extensible end-to-end document processing system for financial documents. From a technical point of view, extracting tiny bits of information from large-volume, financial documents is a highly imbalanced task, with an unrestricted set of rest classes. Given the nature of this needle in a haystack problem and the associated robustness requirements, we show that our method Informer, derived from supervised outlier detection, seamlessly integrates with the other extractor models in this system. Noteworthy, the use case of financial document processing is representative for any document processing pipeline, and thus, the insights translate to any other system involving domain-specific extractors (e.g., based on ATA, DAE, and Informer). In this chapter, we outline the entire system including the concrete design choices, the architecture specification, and the algorithmic realization of the extractors. Finally, we

provide in-depth analysis on the scalability and extensibility of the system.

This chapter is based on our publications [202, 222]. The idea in [202], to apply outlier detection methods to toxicity detection, was proposed by Max Lübbering and extensively discussed with all co-authors, who worked on similar research questions at that time. The experiments of this paper were mostly conducted by Max Lübbering. The writing and revising of the paper was shared between all authors, while Max Lübbering contributed the most significant part.

The publication [222] was the result of a customer project at Fraunhofer IAIS, which was lead by Max Lübbering. The models were implemented by the first six co-authors in the author list. The architectural design of the document processing pipeline was mainly proposed by Max Lübbering with the support from his co-authors. The idea to subsequently integrate robust classification methods such as Informer was proposed, implemented, and evaluated by Max Lübbering.

6.1 Toxicity Detection in Online Comments with Limited Data: A Comparative Analysis

The steadily increasing amount of online communication has been rendering manual moderation almost infeasible. This affirms the importance of automatic detection of toxic content (related to, e.g., cyber bullying and harassment [223]) in online conversations. There are different types of toxic comments that are commonly observed, such as threats, insults or attacks based on people's race and sexual orientation. An effective system for toxicity detection should be able to detect all of them with a high accuracy, and even generalize to unseen toxicity types, while not wrongly classifying normal comments as toxic.

Toxicity classification poses a supervised classification problem whose existing solutions can be broadly categorized into two categories [224]: manual feature engineering, and deep learning methods. While in the first case, features are manually selected and fed to the classifier as input vectors, deep learning approaches aim to learn seemingly abstract features present in the text on their own.

A key problem in solving this issue with deep learning, is that there are often no sufficient amounts of data available for all toxicity types. While conventional machine learning systems are very accurate in correctly identifying common types of toxicity, such as curse words or obscene language [224], they might lack generalization by failing at detecting other, less obvious attacks.

In order to address this issue of diverse and previously unknown toxicity types, we present a comparative analysis of classification and outlier detection methods. We specifically investigate each system in challenging, but very common, settings by a) downsizing the training sets and b) constraining the training set to a single type of toxicity, utilizing the remaining classes solely for evaluation. This setup therefore enables us to directly measure the generalization and robustness performance of the algorithms.

In this work, we consider three different types of methods for toxicity detection, namely a) representation learning based outlier detectors via ATA (see Sec. 3.5), b) ensemble methods and c) traditional deep neural networks. In the first case, a representation of the class of interest (COI) / inliers, i.e., the toxic class, is learned and any sample that is very dissimilar from this representation is being rejected as an outlier [23, 24, 64, 67]. Note that, due to the large and unrestricted manifold of normal communication, which is infeasible to learn apart from large-scale language models, we decided to learn the manifold of toxic communication. Specifically, ATA is composed of an autoencoder that predicts the reconstruction error for a given sample. Due to a custom training approach that

maximizes / minimizes the reconstruction loss for outliers and COI, respectively, the reconstruction error becomes highly predictive of the outlierness of a sample, as shown in the previous chapters. As a second baseline based on representation learning, we consider *one class autoencoders* (OCA), a semi-supervised method, which in contrast to ATA only minimizes the reconstruction error of COI samples.

Similar to aforementioned outlier detectors, deep learning based ensemble methods have been proven to be more robust than plain MLPs [35, 174]. In this work, we specifically consider the MIMO [174] architecture as an ensemble representative, which incorporates the ensembling in a single neural network. Due to this, MIMO makes more efficient usage of parameters, and is less overparameterized compared to MLPs [174]. Finally, to put the baseline performances into perspective, we also consider an MLP, one of the most classic methods for binary classification.

Our contributions can be summarized as follows:

- We present a custom experiment setup by limiting the training set size and constraining the observed toxicity types. This setup enables us to evaluate the models as close to real-world scenarios as possible.
- We compare methods from three different areas, namely representation learning, ensemble methods, and deep learning methods solely optimized for classification.
- Our evaluation on the toxicity detection task comprises three different aspects: Classification performance, generalization capabilities, and robustness.

6.1.1 Toxicity Detection Dataset

In this work, we use the toxicity detection dataset published by Google Jigsaw for the Toxic Comment Classification Challenge [225] on Kaggle. This multi-label dataset originally contains 159,751 training samples and 153,164 independent test samples. The samples have been annotated by 5000 human annotators according to their toxicity level. These annotated comments were categorized into six toxicity classes: *toxic*, *severe toxic*, *insult*, *threat*, *obscene* and *identity hate*. Note that toxic comments can belong to more than one toxic class. In fact, only 39.2% of them have been categorized with just one label.

For better interpretability of the results and prevention of information leakage, we define an additional label, *toxic-only*, which is assigned to those samples that have only the *toxic* category annotated (and no other toxicity label).

For our experiments, we consider a strongly reduced version of the original data set. This is done to simulate the common situation, where only limited data is available, and to make it harder for the algorithms to learn general properties of the data. Firstly, we remove all samples from the training split that have any label other than *non-toxic* and *toxic-only*. Secondly, we apply downsampling to further reduce the overall dataset size. To make the dataset suitable for binary classification, we treat all comments with any toxic label as *toxic*, and all others as *non-toxic*.

The evaluation is done on four separate test sets. They all contain the same, randomly sampled 10000 non-toxic samples from the original test split, and a number of toxic samples with distinct types. They are defined as specified in Table 6.1. Note that we allow for overlap with different toxic labels, except for the *toxic-only* test split, which consists of samples with only the *toxic* category (see above). As part of the preprocessing, all samples were represented as pooled Glove word embeddings.

Test split	#toxic samples	#non-toxic samples
toxic-only	1710	10000
threat	654	10000
insult	10686	10000
identity-hate	1995	10000

Table 6.1: Number of toxic samples for each of the four test splits. Note, that each test split shares the same 10000 non-toxic samples.

6.1.2 Experiments

For each method, we apply an extensive grid search (GS) over multiple parameter settings. We perform hyperparameter-tuning w.r.t. *learning rate*, and *weight decay* for each method, and specifically w.r.t. *outlier weighting factor* and *outlier bin start* for ATA.

To achieve a fair comparison, each model is parameterized with comparable complexity. The MLP has four hidden layers of sizes 100, 50, 25 and 12 and a binary output. MIMO has an ensemble size of 3 and hidden layers of size 50, 25 and 12. Finally, ATA comprises three hidden layers of sizes 60, 30, and 15 for the encoder, as well as for the decoder, albeit in reverse order. All methods have sigmoid activations. In conclusion MIMO, MLP and ATA have 16650, 16765, and 16750 trainable parameters, respectively.

As mentioned in the beginning of Sec. 6.1, we chose to minimize the reconstruction error of *toxic* samples for the representation learning methods (ATA and OCA) because we find that the models are able to generalize better using this setup. Intuitively, toxic comments tend to share a rather limited vocabulary and range of topics, which is why they are more homogeneous among each other in comparison to *non-toxic* comments with their gigantic manifold.

6.1.3 Results

For the evaluation, we consider the AUPR [171] and F1 score, both w.r.t. the *toxic* class. AUPR is a threshold-independent metric, which takes the base rate of the positive class into account. Since the toxicity dataset is highly imbalanced, this metric yields estimations that reflect the expected performance at deployment time. We also report the F1 score to measure the model performance w.r.t. classification and reasonable threshold learning.

As shown in Table 6.2, the MLP, which is solely optimized for classification, does not generalize well to unseen toxicity types. While the overall classification performance on *toxic-only* is close to MIMO, which is the best classifier on the toxic-only split, MLP shows significantly higher performance degradation on the unseen toxicity classes such as *threat*. MIMO is the most stable method among the four baselines. It provides strong classification performance on the known, toxic-only class, but also generalizes well to the three unseen toxicity classes. Nevertheless, similar to MLP, we also find that MIMO tends to fail at times, as seen in the *threat* class on the smallest training set. ATA shows the strongest performance on unseen toxicities, while also providing competitive results on the toxic-only split. Interestingly, ATA never yields any complete failures compared to the other baselines, indicating that the representation learning setup achieves the highest robustness. Finally, we also see that the training approach is crucial. While ATA incorporates not only *toxic*, but also *non-toxic* samples in

test split	method	toxic: 250 non-toxic: 1250		toxic: 1000 non-toxic: 5000		toxic: 5000 non-toxic: 60000	
		AUPR	F1 Score	AUPR	F1 Score	AUPR	F1 Score
toxic-only	BASE	14.6	11.3	14.6	11.3	14.6	11.3
	MLP	49.1	46.9	48.5	48.5	51.0	48.6
	MIMO	49.8	50.7	51.0	51.2	53.2	49.7
	ATA	47.7	50.4	44.6	48.0	51.8	52.6
	OCA	14.4	8.5	15.3	9.3	14.4	10.3
threat	BASE	6.1	5.5	6.1	5.5	6.1	5.5
	MLP	50.9	29.5	49.3	48.5	62.9	31.3
	MIMO	47.3	36.3	65.7	36.6	67.9	32.9
	ATA	65.0	38.8	65.2	38.8	67.9	40.4
	OCA	5.7	7.2	7.3	9.8	5.7	7.2
insult	BASE	51.7	25.4	51.7	25.4	51.7	25.4
	MLP	91.9	85.0	91.8	85.6	93.5	86.0
	MIMO	91.9	85.8	92.7	86.7	94.0	86.5
	ATA	92.5	85.6	91.4	83.9	93.4	87.0
	OCA	53.3	14.4	56.7	17.6	53.3	17.6
identity-hate	BASE	16.6	13.5	16.6	13.5	16.6	13.5
	MLP	76.4	55.4	74.6	56.1	81.5	57.3
	MIMO	75.6	62.3	74.5	62.2	82.5	59.1
	ATA	78.4	63.1	76.4	62.0	81.7	64.7
	OCA	16.1	10.4	18.3	12.6	15.9	11.1

Table 6.2: Performance of MLP, MIMO, ATA and OCA on test splits toxic-only, threat, insult and identity-hate. We consider the AUPR and the F1-Score (both with respect to the "toxic" class). As a reference for the base rate dependent metrics, we also report the expected scores for a random classifier (BASE) with uniform probabilities $p \sim U[0, 1]$.

the training procedure, OCA's training routine exposes the model only to toxic samples, leading to underperformance, even compared to the random BASE baseline. This is a well-known problem which especially arises when the inliers correlate with outliers in feature space [23, 24]. Interestingly, all methods have superior performance on the insult test set, compared to the others. This could be explained by the fact that insults are relatively easy to spot based on certain keywords, while threats and identity-related hate are usually context-dependent.

Relevant for the training and deployment of toxicity detection systems, we find that training set size has a significant impact on the generalization performance of such systems. ATA, MIMO, and MLP all improve their AUPR scores with bigger training set sizes. Interestingly, but not unexpectedly, this even transfers to toxicity classes not seen during training.

6.1.4 Challenges of Encoding Toxicities

Despite the competitive classification performance and robustness of ATA in comparison to MLP, MIMO, and OCA, the ATA method has an architectural disadvantage to MLP and MIMO in case of data abundance. As specified by us, reconstructive representation learning for classification forces the model to encode all the information present in the data, including noisy features and the diverse set of topics present in the corpus, irrespective of the relatedness to toxicity. In contrast to ATA, the MLP and MIMO are able to reject such features and topics as uninformative, leading to better performance when the ML problem is sufficiently covered by the data.

The issue of unfiltered information encoding is especially comprehensible when considering applying ATA to the TREC dataset [207]. The task here is to classify topic-wise unrestricted questions regarding the type of question, e.g., if the question asks for an entity or a location. Here, the interrogative word (e.g., what, which, who, etc.) in the questions provides a highly predictive feature that can be directly leveraged via the MLP. On the contrary, ATA has to encode the entire set of features and corpus topics, which, in the limit, essentially translates to having to incorporate the entire world knowledge into its encoding. This limitation can be explained via the manifold hypothesis, outlined in Sec. 1.3. The class of interest samples (here, sentences with a particular interrogative word) lie on such a large manifold that they cannot be embedded by ATA.

These settings can therefore render the reconstruction error less predictive of the inlierness of a sample, when the model's capacity is exceeded. We partially prevented this by treating the toxic samples as the class of interest, and the normal instances as the rest samples. This measure narrowed down the topics within the class of interest, allowing ATA to successfully encode the toxicities and to outperform the MLP. Nevertheless, the results are not as impressive as the open set recognition results provided in Sec. 4.7 for ATA and DAE, and the reasons for this can be attributed to the nature of the toxicity classification task.

Another limitation is caused by the word embedding algorithm GloVe that we used to derive the document-level embeddings. GloVe provides word embeddings based on word co-occurrences and document-level embeddings are obtained via pooling of word embeddings. The resulting embeddings do not reflect the semantic structure of the sentences and therefore lose predictive power on toxicity detection with its predominant contextual dependency.

6.1.5 Conclusion

Toxicity detection is a challenging task. There are various types of toxicity, and naturally, not all types of toxicity can be observed during training. This is why it is inevitable to have algorithms, which are able to learn the abstract toxicity concept and thereby generalize well to unseen toxicity types. Our results show that deep learning methods, which are solely optimized for classification, such as MLPs, lack generalization performance and even tend to fail completely. With ATA and MIMO, we showed that representation learning and ensembling can significantly improve generalization and classification performance. However, the task of encoding toxicities on a low dimensional manifold poses severe difficulties to ATA and OCA, demanding further research. Therefore, as part of future work, we consider applying these approaches to other datasets, to further test their generalization to toxic comments from other social media sources. The insights of this case study suggest an extension of ATA that is capable of filtering and rejecting unimportant features that reduce the complexity of the manifold, matching the complexity of the machine learning problem at hand.

6.2 Deployment Case Study: Automatic Indexing of Financial Documents via Information Extraction

In regulated financial markets, companies are required to disclose relevant information to the public. This information is essential for current and potential investors to make informed decisions regarding their investments. Depending on the use case, there are different types of formal reports, including financial statements, also known as IFRS reports, that inform on the company's finances, and prospectuses that inform potential investors about the opportunities and risks of buying securities issued by the company. Besides these document types, we also consider non-financial reports and adhoc messages. Overall, the information disclosed in these documents comprises crucial details about strategies, fund management, risks of the investment, key performance indicators, etc., in order to inform potential and current shareholders, but also help to protect investors from insider trading.

The vast number of financial documents that are released on a daily basis renders manual filtering and sorting of information impossible. With the advancements in natural language processing, however, automatic document processing becomes feasible and poses a practical solution to get a grip on this flood of information.

Technically, extracting tiny bits of information from large volumes of documents resembles the nature of a needle in a haystack problem. For instance, a model, filtering a single document type, needs to look out for hidden, token-level features characterizing the document type. Moreover, the model is expected to be highly robust to all the possibly existing document types, it might be exposed to at deployment time. Consequently, a document processing system can be associated with open set recognition (OSR) and robust (multi-class) classification, which we introduced in Ch. 4 and Ch. 5, respectively.

To this end, we present an end-to-end document processing system that automatically extracts the essential meta-information of prospectuses and financial statements using techniques of natural language processing and machine learning. Our proposed system consists of several information extractors that extract information such as prospectus type, financial-report type, the issuer name, filing data, etc., which are accessible to clients via a RESTful web interface. In particular, we implement the Informer architecture for the document type classifier. Since the robustness benefits have been sufficiently verified in Ch. 5, and due to lack of sufficiently many annotated prospectuses in the dataset, model robustness is out of the scope of this chapter. Instead, we show via the document type Informer, how our proposed methods can be seamlessly integrated with various extractors into a large-scale, high-volume application, as demonstrated by the proposed document processing pipeline.

We are faced with several challenges in developing such a system, mainly attributed to reasons that are specific to the aforementioned financial documents, which are listed below:

- The documents generally comprise hundreds of pages, and the relevant information is generally found only in few places in the document. This increases the difficulty of information extraction tasks, similar to the problem of finding a needle in a haystack. The limited annotation data, e.g., one entity in an entire document, makes learning within a large hypothesis space extremely difficult. Here, aspects of supervised outlier detection and open-set recognition come into play, due to the severe class imbalance translating into a needle in a haystack problem.
- Due to the large number of documents and their size, the architecture of the system must consider the need for increased computational requirements. Therefore, we opt for a scalable

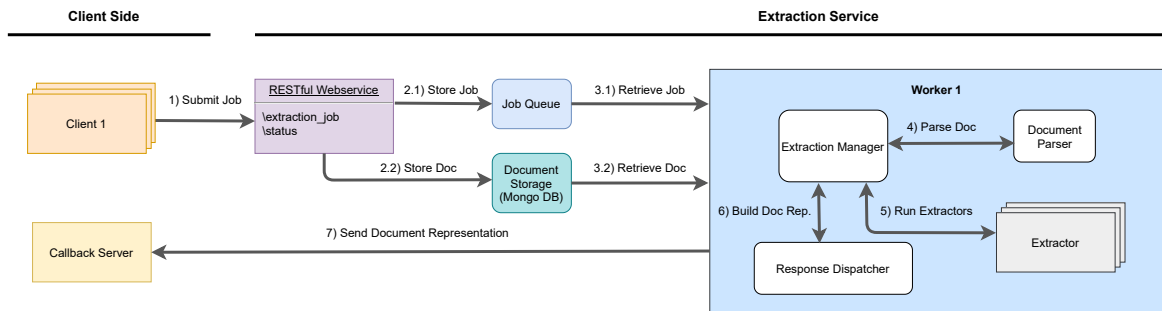


Figure 6.1: Overview of the Extraction Service showing the execution flow of an extraction job through the components in the system. First, a client schedules a document for extraction (1), a job is created and its corresponding PDF is stored (2). A worker then retrieves the job and document (3), parses the document (4), extracts information (5), and builds a JSON representation of the document (6), before sending the document back with its extracted information (7).

solution which allows for horizontal scalability through the use of independent workers.

- These documents are usually issued in PDF-format, which is an unstructured binary format and does not reflect any information w.r.t. ordering and semantics [226]. As one of the effective solutions, we resort to a computer-vision based technique for analyzing layouts as it provides coherent passages for parsing. Therefore, in our setup, due to the non-standardized layout and document structure, we apply a custom training method for the object detection network (ODN) based on the works of [226].

In summary, we present a scalable information extraction system (hereafter referred to as *Extraction Service*) for extracting key meta-information from raw financial documents in PDF format. We show that our solution can support different machine learning models and document types, making the architecture easily extensible and adaptable to new settings (not just limited to the financial domain).

This deployment scenario represents a typical use case of supervised outlier detection, open set recognition, and robust classification, as the goal is to robustly extract a single bit of information within documents comprising hundreds of pages. In this particular use case, we opted for a diverse set of models, including Informer, to stress the architecture’s generalizability.

Related Work

In this section, we provide a short survey of related work concerning financial document processing and analysis.

Traditional offerings such as Yahoo Finance, Google Finance, and Bloomberg Terminal, provide up-to-date information on companies and markets. These systems focus on data such as price history, press releases, and analysis. While these systems are often openly accessible and provide financial (meta-) information on a large scale, the underlying system architecture is not disclosed. In [227], Sifa et al. propose a context-aware recommender system that assists auditors in matching the text in financial statements to legal requirements, using supervised and unsupervised machine learning methods. In [228], Brito et al. present a system that parses published reports on the web and extracts key performance indicators (KPIs) from the parsed text. Additionally, these are made available via a

graphical interface to the users for querying on demand. In a similar work [229], the authors analyze sustainability reports published by companies and extract sentences containing ESG (environmental, social, and governance) indicators. Focusing on extracting named entities from loan agreements, the authors of [230] propose the use of domain adaptation techniques to learn from out-of-domain data. Furthermore, there are multiple application papers concerning financial document processing within the legal / accounting domain [227, 231, 232].

It is noteworthy, that most of the related work concerning natural language processing (NLP) is centered around algorithmic contributions targeted at solving a concrete application task, or to improve state-of-the-art results for prominent NLP tasks [233–235]. These research directions are not within the scope of this chapter. Instead, with the *Extraction Service*, we focus on an architectural contribution; a generic system that is scalable and extensible to any information extraction task from documents, and most importantly, supports dependencies between extractors as part of a dependency graph. We showcase its effectiveness in the use case of metadata extraction from financial documents with its connection to supervised outlier detection, open set recognition, and robust classification.

6.2.1 Extraction Service

The *Extraction Service* is designed to provide a scalable and extensible solution for metadata extraction from financial documents. In the following sections, we describe in detail how we developed this system that caters to these requirements.

Design Choices

To achieve horizontal scalability, we developed our system based on the *producer-consumer* pattern (PCP) [236, 237] with a centralized job queue and distributed consumers. In our setup, *producers* correspond to clients which use our web service and submit extraction jobs; *consumers* correspond to workers, which execute the given job. A processing job always contains a single document and comprises two steps: the parsing, and the meta-information extraction (IE) steps. We model each processing job to be atomic, meaning that they cannot be split across workers. This type of modeling simplifies the processing of a job, as no post-processing step (e.g. aggregation) or synchronization of workers is necessary. Since extractors are dependent on other extractors, treating jobs as non-atomic would add more complexity to the system, which is avoided in our setup. Although it restricts the possible intra-job parallelism (a job is processed in chunks in parallel), we find that a single worker can process a document in a reasonable time.

Since the execution of a job can take several minutes depending on the worker’s hardware specification and length of the given document, we apply the *callback pattern* for asynchronous network communication with the client. This is achieved by allowing the client to provide a callback URL, to which the worker returns the result response with the extracted meta-information.

All of our extractors implement the same interface for extensibility such that new extractors can be easily added or extended in a plug-and-play fashion. It is to be noted that one extractor can depend on other extractors for its processing. For instance, the fiscal year extractor depends on the document type extractor, since fiscal year is to be found only in financial reports; therefore, the document type extractor must be executed before the fiscal year extractor. To tackle this, our extractors are organized in a dependency graph, ensuring that appropriate order of processing the extractors.

System Overview

The *Extraction Service*, as shown in Fig. 6.1, can be divided into four modules: *RESTful Webservice*, *Job Queue*¹, *Document Storage* and *Worker*. The *RESTful Webservice* provides an interface for the clients to submit a processing job (i.e., document along with a callback URL) or to retrieve the status of a submitted job. Upon receiving a job request, the web service adds the job to the *Job Queue* and the respective document to the *Document Storage*. On receiving a job from the *Job Queue*, a *Worker* processes the given document and sends the result back to the client via the provided callback URL.

Within a *Worker*, a given job is processed in two steps: parsing and extraction. First, the given raw PDF document is parsed by the *Document Parser*, which converts it into a structured textual format. Next, the *Extraction Manager* runs the information extractors using the parsed document according to the dependency graph. Finally, the response, which consists of parsed text document along with the extracted key meta-information in JSON format, is sent by the *Response Dispatcher* to the client. This decoupled setup, provides a maximum of flexibility on the client's side for how to proceed with the received data. A typical set up would be to store the data within a document or transactional database, depending on the use case, and then provide a webservice interface which renders the data in a human-readable format, thereby simplifying navigation through the data.

Document Parser

Despite the success of recent neural network architectures such as Faster RCNN [182] in object detection, they provide noisy results when applied to PDF images [226]. Therefore, subsequent post-processing steps are necessary [240–242]. Considering these findings, we base the *Document Parser* on a hybrid-parsing approach that comprises the Faster RCNN architecture for bounding box prediction and DBSCAN [243] / rule-based approach for subsequent fine-tuning, as proposed by [226]. To this end, we trained the Faster RCNN network with 2169 pages of financial documents annotated with bounding boxes having labels *table*, *headline*, *image*, *paragraph* and *header/footer*. Given this network, the raw PDF is split into pages and their corresponding images are fed into the network to obtain bounding boxes. These bounding boxes are then subject to post-processing according to [226]. With the obtained bounding boxes, the text in the document is finally extracted using pdftotext² or Tesseract OCR[244].

Extractors

Next, we turn our attention to the core component of our system, namely the information extractors, that extract key meta-information from prospectuses and financial reports based on the parsed text. Currently, we support the extraction of general attributes of documents such as document language, document type (i.e., prospectus, financial report, non-financial report), and issuer name (company name). Besides these, there are extractors specific to certain types of documents. For prospectuses, extraction of prospectus type (debt, debt supplement), rule 144a (whether Rule 144a³ applies) and filing date are supported. Similarly, from financial reports, information such as financial report type and fiscal year can be extracted.

¹<https://redis.io/>

²<https://poppler.freedesktop.org>

³<https://www.sec.gov/reportspubs/investor-publications/investorpubsrule144htm.html>

6.2 Deployment Case Study: Automatic Indexing of Financial Documents via Information Extraction

Extractor	ML model	Depends on	Prereq.	Description
Language	Naïve Bayes	–	–	Extracts a document’s language. Model is based on [238]
Document Type	Informer	Language	Lang: en	Extracts the document type (prospectus, financial report, non-financial report, and adhoc message) based on keyword appearances and other high-level features such as number of pages. We performed a grid search concerning <i>learning rate</i> and <i>outlier weighting factor</i> .
Prospectus Type	Latent Semantic Analysis (LSA), Logistic Regression	Language, Document Type	Lang: en, Doc. Type: prospectus	Extracts prospectus sub types (debt, debt supplement, equity IPO, equity Non-IPO). The LSA featurizer is parameterized with a dimensionality of 10 and set to 50 iterations
IFRS Report Type	Logistic Regression on keywords	Language, Document Type	Lang: en, Doc. Type: IFRS	Extracts the subtype of an IFRS report(Q1, Half-year, Q3 or Year End)
Rule 144a	Logistic Regression on hand-crafted features	Document Type, Language	Lang: en, Doc. Type: prospectus	Extracts whether Rule 144a applies
Company Name	bi-LSTM for NER, Lookup Table, Fuzzy Sequence Matcher	Language	Lang: en	Extracts the name of the issuing company and maps it to the Legal Entity Identifier (LEI), a 20-character alphanumeric code that uniquely identifies a legal entity participating in financial transactions. The model comprises an NER sequence tagger based on [239], which filters organizational entities that are subsequently filtered by fuzzy matching them against a company lookup table.
Filing Date	Decision Tree with custom feature engineering	Document Type, Language	Lang: en, Doc. Type: prospectus	Extracts the filing date of a prospectus report. The decision tree facilitates gini as the splitting criterion and has a minimal sample split of size 2, which were determined via grid search.
Fiscal Year	Random Forest Regressor	Language, DocumentType	Lang: en, Doc. Type: IFRS	Extracts the fiscal year of ab IFRS report. The random forest regressor comprises 300 trees and uses the mean squared error as a splitting criterion.

Table 6.3: The available extractors with their dependencies and machine learning (ML) approaches. The features for each extractor were manually engineered by accounting domain experts.

Extractor	F1 score	Precision	Recall
Language Extractor	0.93	0.87	0.99
Document Type Extractor	0.75	0.68	0.88
Prospectus Type Extractor	0.93	0.96	0.92
Report Type Extractor	0.98	0.98	0.98
Rule 144a Extractor	0.88	0.88	0.88

Table 6.4: The performance of document-level classifiers measured in terms of F1 score, precision, and recall. The multi-class classifiers are evaluated in terms of the respective macro scores. Note that the imbalance of the document types within the dataset causes a significant drop in F1 score for the document type extractor, due to the macro averaging.

Extractor	Accuracy
LEI Extractor	0.91
Filing Date Extractor	0.92
Fiscal year Extractor	0.94

Table 6.5: The performance of entity extractors in terms of accuracy as a "hit-or-miss" metric.

In general, we can broadly classify the extractors into two categories; *document classifiers*, which assign a class label to a document (e.g., document type and prospectus sub-type) and *entity extractors*, which extract an informational entity out of the text (e.g., name of the issuing company). We employ a variety of machine learning methods to implement these extractors, including Informer (introduced in Ch. 5), logistic regression, decision trees, random forests, and recurrent neural networks. We select the respective algorithm based on different factors such as the availability of annotated training data, the required level of robustness, the complexity of the extraction problem, and feedback from domain experts. Table 6.3 summarizes the available extractors, their dependencies, and underlying ML models.

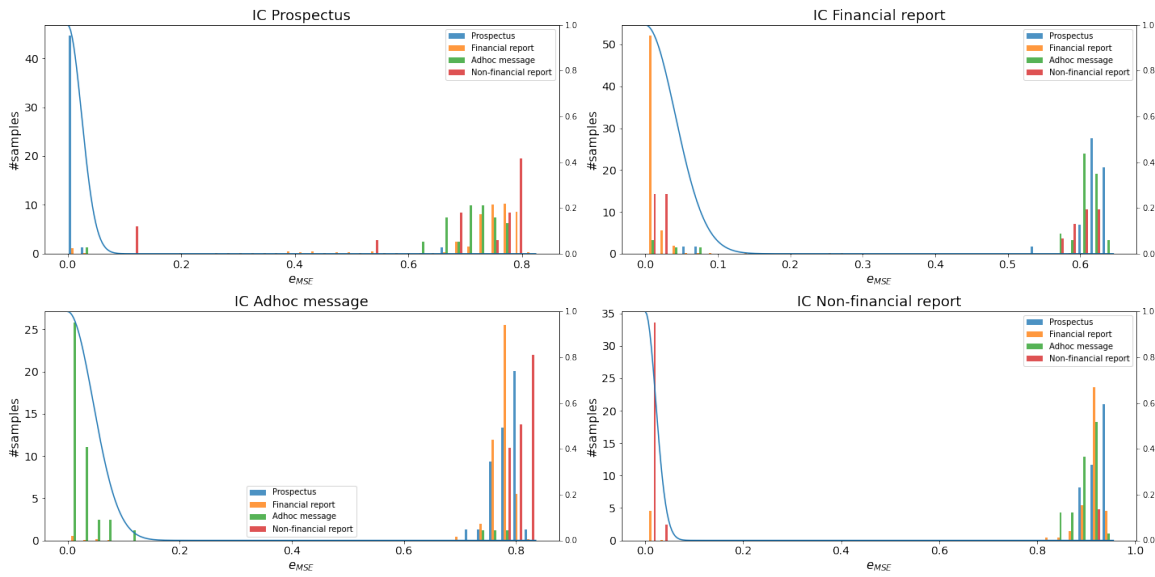
From a software design perspective, each extractor implements a pre-defined interface and thereby acts as a wrapper for any ML model. This not only allows the implementation of different extractors with different ML models, but also provides the flexibility to change or even replace underlying models easily without significant code changes.

6.2.2 Experiments and Results

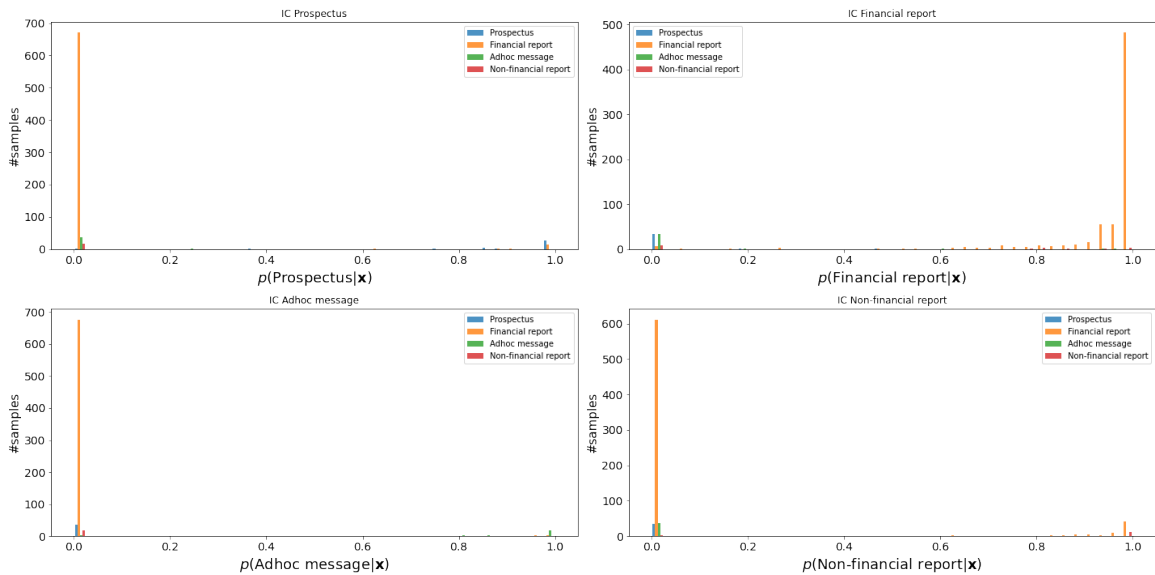
We evaluate the proposed system w.r.t. meta-data retrieval performance of extractors and the scalability of the overall system.

To do this, we trained our models on a partially annotated corpus of 183 prospectuses, 3468 financial statements, 172 adhoc messages, and 73 non-financial reports, having an average number of pages of 294, 78, 2, and 75, respectively. The effective number of documents used to train an extractor varies greatly depending on the number of annotations available for each extraction type. For instance, while there are thousands of document type annotations, there are only 46 for rule 144a. Nevertheless, annotated documents corresponding to each task are split into *train*, *val*, and *test* document sets. For each extraction task, we select the best model settings based on the performance on the *val* split.

6.2 Deployment Case Study: Automatic Indexing of Financial Documents via Information Extraction



(a) Normalized reconstruction error histograms for each IC and for the four classes prospectus, financial report, adhoc message and non-financial report. The blue line shows the gaussian function g , mapping the reconstruction error onto probability scores (see Fig. 5.2 and Eq. 5.1 in Sec. 5.3 for a full derivation). Given exceptionally low support for non-financial reports and adhoc messages, the Informer components still learn generalizing representations for the respective inlier classes.



(b) Unnormalized histograms of probability estimates w.r.t. the different classes and each IC. As already shown in Fig. 6.2(a), the inlier classes are well separated from the respective rest classes in each IC. The class imbalance in the test split of 688 financial reports, compared to 38 prospectuses, 35 adhoc messages, and 17 non-financial reports, leads to a small amount of financial reports being classified as false positives, as can be seen for IC Prospectus and IC Non-financial report.

Figure 6.2: Class-specific prediction histograms for each Informer component (IC). Fig. 6.2(a) shows the normalized (i.e., density) reconstruction error histograms for the four ICs, inferred from the test samples. Likewise, Fig. 6.2(b) shows the estimated probability scores, as unnormalized (i.e., absolute frequencies) histograms. Both diagrams show the robustness of the Informer method and explain that the low precision scores in Table 6.4 is caused due to the class imbalance.

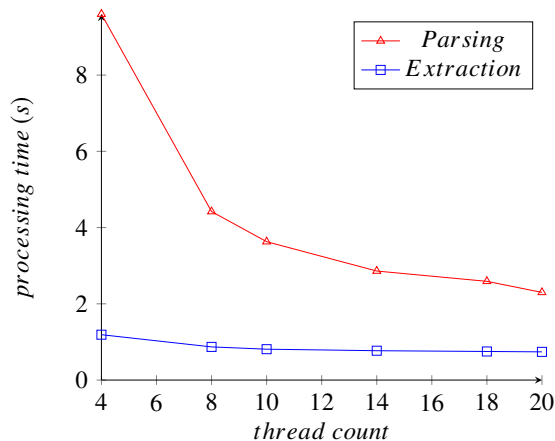


Figure 6.3: The average processing time per page with different thread counts.

We present the performance results of *document classifiers* and *entity extractors* on the *test* split in Table 6.4 and Table 6.5, respectively. We measured the classification performance of the document classifiers w.r.t. F1 score, precision and recall. For the extraction task, the concept of classes does not apply, since we are not classifying but extracting entities. This is why we opted for accuracy as a "hit or miss" type of metric. Despite the difficulty of the task and the low availability of annotated data, our extractors achieve impressive results due to the use of hand-crafted features with the help of domain experts. This is a common approach in outlier detection, as outlined in Ch. 1, preventing overfitting and improving model generalization.

Our results show that almost all methods have a decent classification and extraction performance, being in the 90% range. While the document type classifier yielded only macro F1 score and macro precision score of 75% and 68%, respectively, these values can be attributed to the significant class imbalance in the dataset. Since the dataset contains 89% financial reports, due to plain statistics, a small amount ends up being classified as, e.g., non-financial reports which are represented only by 2% in the test data. As a result, we receive low precision scores for the IC having non-financial reports as inliers, which can be also seen in the histograms in Fig. 6.2. In fact, accounting for the class imbalance by weighted averaging results in a F1 score, recall, and precision of 93%, 91%, and 95%, respectively.

From a design point of view, our document processing architecture is trivially horizontally scalable, allowing throughput to be increased linearly given that there are no bottlenecks within the job dispatching. Nevertheless, parallelism within each task and node is also important. Therefore, we measure parsing and extraction time using varying numbers of threads, as some extraction tasks can benefit from multi-threading. Specifically, models for instance, which leverage the NumPy library [87], immediately benefit from NumPy's multi-threading support. In Fig. 6.3, we present throughput measurements, on a single node with two 2.6 GHz Intel(R) Xeon E5-2697 v3 CPUs having a total of 28 cores. This evaluation shows that our system is also capable of scaling vertically, i.e., in a single node, complementing its horizontal scalability which allows more computationally capable workers to be employed in parallel. Furthermore, it also shows that even modest increases in the number of threads already provide significant speedups. In this manner, meaningful improvements are obtainable with smaller, marginal investments in infrastructure.

6.2.3 Conclusion and Outlook

In this work, we addressed the challenges concerning meta-information extraction from large volumes of financial documents available in PDF format by presenting an efficient system based on machine learning. We found that horizontal scalability and extensibility are essential requirements that should be considered right from the design process. Although our methods, including Informer, are capable of robustly extracting information in a vast amounts of data, their effectiveness is largely determined by their scalability in big data environments. Our experimental results show that our proposed system achieves high throughput and extraction accuracy, using expert knowledge and manual feature engineering. With more labeled data for existing and new meta-data tags, e.g., sourced via online learning, we could implement more sophisticated extractors, e.g., based on *decoupled autoencoders* (see Ch. 4) or Informer, thus establishing an insightful financial tool for investors and researchers with unmatched robustness properties.

Conclusion

The advancement of ML methods into our daily lives, in combination with an increasing number of AI accidents in recent years, has led to rising concerns regarding AI-safety and the reliability of ML models. Today, ML models enhance the images taken with our phones, read the news via intelligent speakers, perform medical diagnosis, and control our cars via autopilot. Similarly, when we engage with the internet, models predict our consumer behavior, recommend videos, and chat with us. Broadly speaking, if there is a business case that can be solved with data, there is already, or will soon be, a model for it. While in many circumstances these models make our lives easier, special precautions must be taken when ML models are deployed environments without regard for safety-critical concerns, potentially harming people, society as a whole or the environment in diverse ways.

In recent years, many researchers have been heavily debating regulations and a codex defining the appropriate contexts for model deployments. In this work, however, we took a purely algorithmic point of view for tackling the prevailing issue of AI-safety. Since DNNs tend to provide overly confident predictions in unknown conditions that are not covered by the distribution of the training data, many AI accidents can be attributed to this unforeseeable behavior of DNNs in the past. We took up this issue and provided a solution to achieve robust binary and multi-class classification by tackling the problem from the perspective of supervised outlier detection. The main benefit of supervised outlier detection lies within the capability of learning a representation of the in-distribution data, allowing intelligent systems to reject significant deviations as unknown. We summarize these steps in the following section and finally provide an outlook on the broader impact of our work including future directions.

7.1 A General Summary

This thesis centers around improving the robustness of DNN-based classifiers. Despite the tremendous achievements of DNNs related to various ML benchmarks in recent years, these methods are still highly susceptible to out-of-distribution data by design. As motivated in the introductory chapter, the application of ML models has led to various AI incidents with human casualties in recent years, stressing the importance of deploying robust DNN models. Before proposing our roadmap towards robust DNN classifiers, we derived the well-established optimization procedure of DNNs and could pinpoint the robustness deficiencies in empirical risk minimization (ERM). Particularly, ERM focuses solely on the separation of the in-distribution classes and misses learning a representation of the in-

distribution data that can be leveraged to reject out-of-distribution data, jeopardizing model robustness. Thus, we introduced (supervised) outlier detection as a viable means to filter out-of-distribution data, sketching out a possible direction towards enhanced DNN robustness.

To this end, we proposed a roadmap comprising three milestones in Sec. 1.5. Each of the milestones listed below is addressed in a single chapter with the goal of making DNNs sensitive to the unknown and encouraging an increase in model robustness:

- Extension of semi-supervised outlier detection towards supervised, deep learning-based outlier detection of application-specific anomalies
- Utilization of supervised outlier detectors for robust one-vs-rest classification in an open-world setting, i.e., open set recognition (OSR)
- Extension of our proposed OSR methods to robust multi-class classification

Before diving into the algorithmic parts in Ch. 3 - Ch. 5, we introduced the infrastructure in Ch. 2 that runs the entire experiment setup and has led to the open-source contributions Datastack for dataset processing and MLgym for reproducible deep learning research.

Datastack provides the data processing pipeline, comprising raw dataset retrieval, preprocessing, and iterator functionality. On an interface level, the data is always handled as byte streams, and only the dataset-specific classes cast the generic byte streams to the concrete data format. As a result, the framework becomes data-format agnostic, supporting any file-format out-of-the-box. Furthermore, DataStack comes with low-level iterator routines such as splitting, merging, and joining that, in combination, can lead to complex iterator functionality.

MLgym is a multipurpose deep learning research framework that emphasizes reproducible ML research. The entire ML setup is formulated within a single config file comprising the Datastack data processing pipeline, trainer, evaluator and model configuration, among others. Generally, there are two alternative approaches on how to design such a system. Commonly, the logic implementation is left to the user, including experiment tracking, training loops, etc., making ML pipelines susceptible to bugs from the start. In our case, we opted for a different solution by providing specific components that can be arranged as a pipeline within the configuration file. The entire pipeline, and each individual component, is fully tested and version controlled, increasing the confidence in the correctness of the experiment results. Besides reproducibility, MLgym supports parallel grid search and (nested) cross validation for large-scale algorithm evaluation. Logging in MLgym is implemented as a distributed system. The experiments can be logged to multiple (remote) servers via websockets, to which clients can then subscribe for certain live experiment events such as metrics or training progress.

In Ch. 3, we explored unknown sensitivity of DNNs from the perspective of outlier detection, which, by design, aims to detect out-of-distribution data. With our adversarially trained autoencoder architecture, we extended the semi-supervised outlier detection method *one-class autoencoder* (OCA) towards supervised outlier detection. Here, the supervision provided a data-inherent description of outlierness and guided the algorithm into the detection of application-specific outliers. We empirically demonstrated its effectiveness by benchmarking ATA against an MLP and OCA, which are potent baselines for classification and outlier detection, respectively. Our results clearly showed ATA's robustness over a range of classification, outlier detection, and novelty detection datasets, while the two other baselines failed on at least one of the tasks. We further explored this direction by utilizing ATA's adversarial training within the multi-task learning architecture *supervised autoencoder* (SAE). Our

results showed similar performance to ATA with a better class separability of the latent representations. With ATA and the SAE variants we provided a well-founded solution to the first milestone.

In Ch. 4, we built upon the insights regarding ATA and proposed the *decoupled autoencoder* (DAE) architecture, an extension of ATA for open set recognition (OSR). In contrast to outlier detection, OSR focuses on the robust detection of a set of classes of interest and disregards any deviations from the in-distribution data. Unlike ATA, DAE learns the decision boundary in an end-to-end fashion, as a tight hull around the inlier data. Here, we could prove that DAE has a bounded open space risk, a key criterion for robust OSR that is not met by the vast majority of OSR methods. Furthermore, we empirically showed that DAE’s objective minimizes the open space risk, providing the highest robustness scores over a large-scale experiment setup, beating various state-of-the-art OSR methods. DAE represents a viable solution to the second milestone.

In Ch. 5, we transitioned from the robust binary classification task explored in the previous chapter towards robust multi-class classification as the final step of teaching DNNs awareness of the unknown, as defined by the third milestone. Technically, we arranged as many DAEs within an ensemble as there are inlier classes, each learning a different one-vs-rest relationship. Our insights clearly demonstrated that the resulting Informer architecture not only inherits the robustness benefits of the DAE architecture, but also allows detecting the two different uncertainty types of aleatoric and epistemic uncertainty. The sources of the two uncertainties are fundamentally different and call for different actions, making this a crucial model characteristic for enhanced AI-safety.

Even though the autoencoder-based methods achieve tremendous robustness gains, they also impose certain requirements on the settings. Our proposed methods aim to reduce/maximize the reconstruction error of the classes of interest / out-of-distribution data, respectively, to leverage the reconstruction error as a predictor for the inlierness of a sample. From a practical point of view, this assumes that the classes of interest already cluster in the feature space to some degree, and that the level of noise and redundant features are nominal. From a theoretical, manifold hypothesis point of view, the input data must lie on a natural manifold that is learnable by the autoencoder. Given that these requirements for the classification problem are fulfilled, then our proposed methods become highly effective. Our toxicity detection case study in Ch. 6 on online communication empirically verified these requirements towards ATA and exposed ATA’s limitations when the natural manifold is difficult to learn.

Deploying outlier detection methods often requires processing large volumes of data to detect an outlier or extract tiny bits of information of interest. In this case, a plain algorithmic contemplation is not sufficient and special attention needs to be paid to the deployment environment to match the scalability requirements. To this end, we proposed a scalable and extensible architecture in Ch. 6 as part of a deployment case study in which we automatically indexed large volumes of financial documents via information extraction. We showed that information extraction can involve hierarchies and dependencies, and engineered an end-to-end document processing pipeline that contains our Informer architecture and scales horizontally on-demand.

7.2 Outlook

Model robustness and, more generally, AI-safety is a research area of high momentum, and new approaches for enhancing model robustness and exposing DNN robustness deficits are being published frequently. With our autoencoder-based methods, we have provided potent algorithms that are sensitive to the unknown, and even determine the level and type of uncertainty within a prediction. Nevertheless,

the overall research field is far from being grazed, and also our own autoencoder-based methods would benefit significantly from further research.

The biggest limitation of ATA, DAE, and Informer is their necessity of encoding and decoding all features, irrespective of their inlier predictability and level of noise. Thus, there are circumstances that impair these methods' performance. When the noise of a few input features exceeds the amplitude of the remaining features (e.g., salt and pepper noise) or the inlier data is spread close to uniformly over the feature space, these methods express difficulties minimizing the reconstruction error to a level that is predictive of the inlierness of a sample. This limitation can be associated with the model's inability to learn the natural manifold of the input data, or a non-existing manifold. In this case, two directions can be considered as a solution. Firstly, offline preprocessing routines can reduce the noise level and compress the features into clusterable representations. Secondly, upstream feature extractors, possibly trained in an end-to-end fashion, could filter the relevant features. Although both preprocessing steps could improve the classification performance, they also might disregard features that are discriminatory for unobserved outliers. A promising trade-off would be to employ distance-aware preprocessors, preserving the feature space structure. Similar ideas have already been applied to DNN methods by, e.g., regularization of the bi-Lipschitz constant [204, 220] and combining spectral normalization on the network weights [245] with neural Gaussian processes for distance awareness [221].

As future work for our open-source contributions, Datastack and MLgym, we would like to lift the two projects from a research-oriented prototype to a production-ready product, supporting larger teams in the development of ML research pipelines, and assisting in conducting reproducible ML experiments. To this end, we plan to support different, enterprise-level storage solutions for binary streams in Datastack, such as MongoDB, Amazon S3, and MinIO, for large-scale, distributed dataset management. Furthermore, we consider implementing further, higher-level iterators, allowing even more sophisticated dataset pipelines. Regarding MLgym, since the general functionality is already advanced and covers the majority of ML research use cases, we would like to improve the scalability instead. One interesting direction is distributed model training, where a single model is split over multiple server nodes, each comprising several GPUs, allowing the training of state-of-the-art large language models, such as GPT-3. Another direction is distributed grid searches across several computation nodes. Both use cases require intelligent bookkeeping of resource allocation, training progress and special attention to fault tolerance.

In conclusion, the algorithmic contributions and open-sourced infrastructure tools in this thesis, have brought novel insights and deep learning architectures to the fields of AI safety, trustworthy deep learning, outlier detection, as well as reproducible deep learning research, pushing state-of-the-art a step forward towards teaching deep learning methods sensitivity to the unknown. Furthermore, we connected these fields in an intelligent manner that paves a possible way for future work on the intersection of these areas.

Appendix

A.1 From Open Set Recognition Towards Robust Multi-class Classification

A.1.1 Derivation of Informer Component Interdependence Induced by Softmax

Let $y \in c$ where $c = \{1, \dots, k\}$ is the set of classes and y, k denote the target and number of classes, respectively. Let $\varphi(\mathbf{x}; \Theta, \sigma_1, \dots, \sigma_k)$ be the Informer network, with model parameters Θ , and decision boundary parameters σ . We calculate the gradient w.r.t. parameter σ_i and σ_j with $i \neq j$, given sample \mathbf{x} of class $y = i$ as input to the Informer network. Considering the σ_i and σ_j separately, we can determine the influence of inlier and rest classes on σ in isolation. Firstly, we calculate the loss gradient w.r.t. σ_i

$$\frac{\partial}{\partial \sigma_i} L_{\text{CE}}(y, \varphi(\mathbf{x}; \Theta, \sigma)) \quad (\text{A.1})$$

$$= \frac{\partial}{\partial \sigma_i} -\log \left(\text{softmax}_i(\varphi(\mathbf{x}; \Theta, \sigma)) \right) \quad (\text{A.2})$$

$$= -\frac{\partial}{\partial \sigma_i} \log \left(\frac{e^{\varphi_i(\mathbf{x}; \Theta_i, \sigma_i)}}{\sum_{n=1}^k e^{\varphi_n(\mathbf{x}; \Theta_n, \sigma_n)}} \right) \quad (\text{A.3})$$

$$= -\frac{\sum_{n=1}^k e^{\varphi_n(\mathbf{x}; \Theta_n, \sigma_n)} \frac{\partial}{\partial \sigma_i} e^{\varphi_i(\mathbf{x}; \Theta_i, \sigma_i)}}{e^{\varphi_i(\mathbf{x}; \Theta_i, \sigma_i)} \sum_{n=1}^k e^{\varphi_n(\mathbf{x}; \Theta_n, \sigma_n)}} \quad (\text{A.4})$$

$$= -\frac{\sum_{n=1}^k e^{\varphi_n(\mathbf{x}; \Theta_n, \sigma_n)} \left(\sum_{n=1}^k e^{\varphi_n(\mathbf{x}; \Theta_n, \sigma_n)} \right) \frac{\partial}{\partial \sigma_i} e^{\varphi_i(\mathbf{x}; \Theta_i, \sigma_i)} - e^{\varphi_i(\mathbf{x}; \Theta_i, \sigma_i)} \frac{\partial}{\partial \sigma_i} e^{\varphi_i(\mathbf{x}; \Theta_i, \sigma_i)}}{e^{\varphi_i(\mathbf{x}; \Theta_i, \sigma_i)} \left(\sum_{n=1}^k e^{\varphi_n(\mathbf{x}; \Theta_n, \sigma_n)} \right)^2} \quad (\text{A.5})$$

$$= -\frac{\sum_{n \in \{0, \dots, k\} \setminus \{i\}} e^{\varphi_n(\mathbf{x}; \Theta_n, \sigma_n)} \frac{\partial \varphi(\mathbf{x}; \Theta, \sigma)}{\partial \sigma_i}}{\sum_{n \in \{0, \dots, k\}} e^{\varphi_n(\mathbf{x}; \Theta_n, \sigma_n)}} \quad (\text{A.6})$$

$$= -(1 - p(c_i | \mathbf{x})) \frac{\partial \varphi(\mathbf{x}; \Theta, \sigma)}{\partial \sigma_i} \quad (\text{A.7})$$

Secondly, let $j \neq i$. With this, the gradient calculates to

$$\frac{\partial}{\partial \sigma_j} L_{\text{CE}}(y, \varphi(\mathbf{x}; \Theta, \sigma)) \quad (\text{A.8})$$

$$= \frac{\partial}{\partial \sigma_j} - \log \left(\text{softmax}_i(\varphi(\mathbf{x}; \Theta, \sigma)) \right) \quad (\text{A.9})$$

$$= - \frac{\partial}{\partial \sigma_j} \log \left(\frac{e^{\varphi_i(\mathbf{x}; \Theta_i, \sigma_i)}}{\sum_{n=1}^k e^{\varphi_n(\mathbf{x}; \Theta_n, \sigma_n)}} \right) \quad (\text{A.10})$$

$$= - \frac{(\sum_{n=1}^k e^{\varphi_n(\mathbf{x}; \Theta_n, \sigma_n)}) e^{\varphi_i(\mathbf{x}; \Theta_i, \sigma_i)}}{e^{\varphi_i(\mathbf{x}; \Theta_i, \sigma_i)}} \frac{\partial}{\partial \sigma_i} \frac{1}{\sum_{n=1}^k e^{\varphi_n(\mathbf{x}; \Theta_n, \sigma_n)}} \quad (\text{A.11})$$

$$= - \frac{(\sum_{n=1}^k e^{\varphi_n(\mathbf{x}; \Theta_n, \sigma_n)}) e^{\varphi_i(\mathbf{x}; \Theta_i, \sigma_i)}}{e^{\varphi_i(\mathbf{x}; \Theta_i, \sigma_i)}} \frac{\frac{\partial}{\partial \sigma_i} e^{\varphi_i(\mathbf{x}; \Theta_i, \sigma_i)}}{(\sum_{n=1}^k e^{\varphi_n(\mathbf{x}; \Theta_n, \sigma_n)})^2} \quad (\text{A.12})$$

$$= \frac{e^{\varphi_j(\mathbf{x}; \Theta, \sigma)}}{\sum_i^{\{0, \dots, k\}} e^{\varphi_i(\mathbf{x}; \Theta, \sigma)}} \frac{\partial \varphi(\mathbf{x}, \Theta, \sigma)}{\partial \sigma_j} \quad (\text{A.13})$$

$$= p(c_j | \mathbf{x}) \frac{\partial \varphi(\mathbf{x}, \Theta, \sigma)}{\partial \sigma_j} \quad (\text{A.14})$$

A.1.2 Derivation of Uncertainty Computation Within Uncertainty Estimation Module

Given prediction $\hat{y} = (p(c_i | \mathbf{x}), p(c_j | \mathbf{x}))^T$, where i, j denote the indices of the two predictions with the highest confidence, the aleatoric uncertainty U_a , epistemic uncertainty U_e , and total uncertainty U_t , are determined as follows.

Let $p(c_i | \mathbf{x})$ and $p(c_j | \mathbf{x})$ span a two-dimensional xy-coordinate system with $p(c_j | \mathbf{x})$ denoting the y-axis without loss of generality. U_t is defined as the relative distance to $(1, 0)$, resulting in any prediction on the line through \hat{y} parallel to the diagonal having an equal U_t .

The x-intercept x_0 and y-intercept y_0 of the parallel computes to

$$x_0 = -\hat{y}_2 + \hat{y}_1 \quad (\text{A.15})$$

$$y_0 = \hat{y}_2 - \hat{y}_1, \quad (\text{A.16})$$

respectively.

To calculate the distance s_t from the parallel to $(1, 0)$ it follows

$$s_t^2 + s_t^2 = (1 - x_0)^2 \quad (\text{A.17})$$

$$s_t = \sqrt{\frac{(1 - x_0)^2}{2}} \quad (\text{A.18})$$

$$s_t = \sqrt{\frac{(1 + \hat{y}_2 - \hat{y}_1)^2}{2}} \quad (\text{A.19})$$

Since the distance between the diagonal and $(1, 0)$ amounts to $\sqrt{\frac{1}{2}}$, the total uncertainty computes to

$$U_t = \frac{s_t}{\frac{1}{2}} \quad (\text{A.20})$$

$$= \frac{\sqrt{\frac{(1+\hat{y}_2-\hat{y}_1)^2}{2}}}{\frac{1}{2}} \quad (\text{A.21})$$

U_a is the distance s_a between \hat{y} and x_o , normalized by the length of the parallel, which amounts to $\sqrt{2(1+y_0)^2}$.

$$s_a = \|\hat{y} - (x_0, 0)^T\|_2 \quad (\text{A.22})$$

$$= \hat{y}_2 \sqrt{2} \quad (\text{A.23})$$

$$U_a = \frac{\hat{y}_2 \sqrt{2}}{\sqrt{2(1+\hat{y}_2-\hat{y}_1)^2}} \quad (\text{A.24})$$

Since U_t is a convex combination of U_a and U_e , it follows $U_e = 1 - U_a$.

A.2 MLgym exemplary pipeline configuration

Listing 2 shows an exemplary deep learning pipeline configuration, including the processing of the MNIST dataset, model specification, training routine, evaluation routine. The *global config* defines the keys that are reused as references throughout the configuration. The *dataset iterator* instantiates the MNIST dataset by subscribing to the MNIST dataset in the *dataset repository*. The *splitted dataset iterators* component splits the *dataset iterator* into the train, val, and test splits. These splits and the *data collator*, which shapes the raw input into the correct format for the neural network, are passed to the *data loaders* component. We create a convolutional neural network, by subscribing to the *model registry* in the *model* component and instantiate the respective model class with the specified model parameters.

The *trainer* and *train component* subscribe to the *loss function registry*, *prediction postprocessing registry*, *loss function registry*, *model*, and *data loaders* to instantiate the loss function, postprocessing of raw predictions, and set up the training routine.

Similarly, the *evaluator* and *eval component* subscribe to the *model*, *data loaders*, *loss function registry*, *metric registry*, and *prediction postprocessing registry* to instantiate the losses, metrics, and postprocessing functionality, and set up the evaluation routine.

Besides the dependency graph specification, the YAML configuration also allows for defining the hyperparameter sweeps, as can be seen for the learning rate parameter in the *optimizer*.

```

1 global_config:
2   storage_connector_path: &storage_path_anchor ./file_storage/
3   seed: &seed_value 2

```

```
4   target_key: &target_key_anchor target_key
5   model_prediction_key: &model_prediction_key_anchor model_prediction_key
6   postprocessing_argmax_key: &postprocessing_argmax_key_anchor
   ↪ postprocessing_argmax_key
7
8   dataset_repository:
9     component_type_key: DATASET_REPOSITORY
10    variant_key: DEFAULT
11    config:
12      storage_connector_path: *storage_path_anchor
13
14   dataset_iterators:
15     component_type_key: DATASET_ITERATORS
16     variant_key: DEFAULT
17     requirements:
18       - name: repository
19         component_name: dataset_repository
20     config:
21       dataset_identifier: mnist
22       split_configs:
23         - split: train
24         - split: test
25
26   splitted_dataset_iterators:
27     component_type_key: SPLITTED_DATASET_ITERATORS
28     variant_key: RANDOM
29     requirements:
30       - name: iterators
31         component_name: dataset_iterators
32       subscription:
33         - train
34         - test
35     config:
36       split_configs:
37         train:
38           train: 0.7
39           val: 0.3
40       seed: 2
41
42   data_collator:
43     component_type_key: DATA_COLLATOR
44     variant_key: DEFAULT
45     config:
46       collator_type:
47       injectable:
```

```
48     id: id_conv_mnist_standard_collator
49 collator_params:
50     target_publication_key: *target_key_anchor
51
52 data_loaders:
53     component_type_key: DATA_LOADER
54     variant_key: FUTURE
55     requirements:
56     - name: iterators
57       component_name: splitted_dataset_iterators
58       subscription: [train, val, test]
59     - name: data_collator
60       component_name: data_collator
61     config:
62     batch_size: 50
63     sampling_strategies:
64     train:
65     strategy: WEIGHTED_RANDOM
66     seed: 10
67     val:
68     strategy: IN_ORDER
69     seed: 10
70     test:
71     strategy: RANDOM
72     seed: 10
73
74 model_registry:
75     component_type_key: MODEL_REGISTRY
76     variant_key: DEFAULT
77
78 model:
79     component_type_key: MODEL
80     variant_key: DEFAULT
81     requirements:
82     - name: model_registry
83       component_name: model_registry
84       subscription: conv_net
85     config:
86     model_definition:
87     layer_config:
88     - params:
89     in_channels: 1
90     kernel_size: 3
91     out_channels: 32
92     stride: 1
```

```
93     type: conv
94   - params:
95     in_channels: 32
96     kernel_size: 3
97     out_channels: 64
98     stride: 1
99   type: conv
100  - params:
101    in_features: 9216
102    out_features: 128
103  type: fc
104  - params:
105    in_features: 128
106    out_features: 10
107  type: fc
108  prediction_publication_keys:
109    prediction_publication_key: *model_prediction_key_anchor
110  seed: *seed_value
111
112  optimizer:
113    component_type_key: OPTIMIZER
114    variant_key: DEFAULT
115    config:
116      optimizer_key: ADAM
117      params:
118        lr:
119          sweep: absolute
120          values: [0.01, 0.001, 0.0001]
121
122  loss_function_registry:
123    component_type_key: LOSS_FUNCTION_REGISTRY
124    variant_key: DEFAULT
125
126  metric_registry:
127    component_type_key: METRIC_REGISTRY
128    variant_key: DEFAULT
129
130  prediction_postprocessing_registry:
131    component_type_key: PREDICTION_POSTPROCESSING_REGISTRY
132    variant_key: DEFAULT
133
134  train_component:
135    component_type_key: TRAIN_COMPONENT
136    variant_key: DEFAULT
137    requirements:
```

```
138     - name: loss_function_registry
139       component_name: loss_function_registry
140     - name: prediction_postprocessing_registry
141       component_name: prediction_postprocessing_registry
142   config:
143     loss_fun_config:
144       prediction_subscription_key: *model_prediction_key_anchor
145       target_subscription_key: *target_key_anchor
146       key: CrossEntropyLoss
147       tag: cross_entropy_loss
148
149   trainer:
150     component_type_key: TRAINER
151     variant_key: DEFAULT
152     requirements:
153       - name: train_component
154         component_name: train_component
155       - name: model
156         component_name: model
157         subscription: null
158       - name: data_loaders
159         component_name: data_loaders
160         subscription: train
161
162   eval_component:
163     component_type_key: EVAL_COMPONENT
164     variant_key: DEFAULT
165     requirements:
166       - name: model
167         component_name: model
168         subscription: null
169       - name: data_loaders
170         component_name: data_loaders
171         subscription: [train, val, test]
172       - name: loss_function_registry
173         component_name: loss_function_registry
174       - name: metric_registry
175         component_name: metric_registry
176       - name: prediction_postprocessing_registry
177         component_name: prediction_postprocessing_registry
178   config:
179     cpu_target_subscription_keys:
180       - *target_key_anchor
181     cpu_prediction_subscription_keys:
182       - *postprocessing_argmax_key_anchor
```

```
183     - *model_prediction_key_anchor
184 post_processors_config:
185     - key: "ARG_MAX"
186       prediction_subscription_key: *model_prediction_key_anchor
187       prediction_publication_key: *postprocessing_argmax_key_anchor
188 train_split_name: train
189 metrics_config:
190     - key: F1_SCORE
191       params:
192         average: macro
193         prediction_subscription_key: *postprocessing_argmax_key_anchor
194         target_subscription_key: *target_key_anchor
195         tag: F1_SCORE_macro
196     - key: PRECISION
197       params:
198         average: macro
199         prediction_subscription_key: *postprocessing_argmax_key_anchor
200         target_subscription_key: *target_key_anchor
201         tag: PRECISION_macro
202     - key: RECALL
203       params:
204         average: macro
205         prediction_subscription_key: *postprocessing_argmax_key_anchor
206         target_subscription_key: *target_key_anchor
207         tag: RECALL_macro
208 loss_funs_config:
209     - prediction_subscription_key: *model_prediction_key_anchor
210       target_subscription_key: *target_key_anchor
211       key: CrossEntropyLoss
212       tag: cross_entropy_loss
213
214 evaluator:
215   component_type_key: EVALUATOR
216   variant_key: DEFAULT
217   requirements:
218     - name: eval_component
219     component_name: eval_component
```

Listing 2: Exemplary deep learning pipeline configuration

Bibliography

- [1] L. Le, A. Patterson and M. White, “Supervised autoencoders: Improving generalization performance with unsupervised regularizers”, *Advances in Neural Information Processing Systems*, 2018 (cit. on pp. 4, 24, 43, 54, 55, 59).
- [2] F. Rosenblatt, *The perceptron: a probabilistic model for information storage and organization in the brain.*, *Psychological review* **65** (1958) (cit. on p. 9).
- [3] D. E. Rumelhart, G. E. Hinton and R. J. Williams, *Learning representations by back-propagating errors*, *nature* **323** (1986) (cit. on pp. 9, 33).
- [4] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, *Gradient-based learning applied to document recognition*, *Proceedings of the IEEE* **86** (1998) (cit. on p. 9).
- [5] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann et al., *Palm: Scaling language modeling with pathways*, arXiv preprint arXiv:2204.02311 (2022) (cit. on pp. 9, 40).
- [6] C.-Y. Wang, A. Bochkovskiy and H.-Y. M. Liao, “Scaled-yolov4: Scaling cross stage partial network”, *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, 2021 (cit. on p. 9).
- [7] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen and I. Sutskever, “Zero-shot text-to-image generation”, *International Conference on Machine Learning*, PMLR, 2021 (cit. on p. 9).
- [8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., *Language models are few-shot learners*, *Advances in neural information processing systems* **33** (2020) (cit. on p. 9).
- [9] M. Shoyebi, M. Patwary, R. Puri, P. LeGresley, J. Casper and B. Catanzaro, *Megatron-lm: Training multi-billion parameter language models using model parallelism*, arXiv preprint arXiv:1909.08053 (2019) (cit. on p. 9).
- [10] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805 (2018) (cit. on pp. 9, 10).
- [11] I. Yamada, A. Asai, H. Shindo, H. Takeda and Y. Matsumoto, *Luke: deep contextualized entity representations with entity-aware self-attention*, arXiv preprint arXiv:2010.01057 (2020) (cit. on p. 9).

- [12] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al., *Mastering the game of Go with deep neural networks and tree search*, *nature* **529** (2016) (cit. on p. 9).
- [13] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko et al., *Highly accurate protein structure prediction with AlphaFold*, *Nature* **596** (2021) (cit. on p. 9).
- [14] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona and T. Blaschke, *The rise of deep learning in drug discovery*, *Drug discovery today* **23** (2018) (cit. on p. 9).
- [15] S. M. McKinney, M. Sieniek, V. Godbole, J. Godwin, N. Antropova, H. Ashrafiyan, T. Back, M. Chesus, G. S. Corrado, A. Darzi et al., *International evaluation of an AI system for breast cancer screening*, *Nature* **577** (2020) (cit. on p. 9).
- [16] A. S. Ahuja, *The impact of artificial intelligence in medicine on the future role of the physician*, *PeerJ* **7** (2019) (cit. on p. 10).
- [17] I. Goodfellow, Y. Bengio, A. Courville and Y. Bengio, *Deep learning*, vol. 1, 2, MIT press Cambridge, 2016 (cit. on pp. 10, 13, 14, 71).
- [18] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman and D. Mané, *Concrete Problems in AI Safety*, arXiv preprint (2016) (cit. on pp. 10, 14, 64, 90).
- [19] K. R. Varshney and H. Alemzadeh, *On the safety of machine learning: Cyber-physical systems, decision sciences, and data products*, *Big data* **5** (2017) (cit. on pp. 10, 90).
- [20] D. Hendrycks, N. Carlini, J. Schulman and J. Steinhardt, *Unsolved problems in ml safety*, arXiv preprint arXiv:2109.13916 (2021) (cit. on p. 10).
- [21] S. McGregor, *Preventing Repeated Real World AI Failures by Cataloging Incidents: The AI Incident Database*, arXiv preprint (2020) (cit. on pp. 10, 90).
- [22] M. Lübbering, M. Gebauer, R. Ramamurthy, C. Bauckhage and R. Sifa, “Decoupling Autoencoders for Robust One-vs-Rest Classification”, *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, 2021 1 (cit. on pp. 10, 11, 30, 63, 90, 92, 94–96, 98).
- [23] M. Lübbering, R. Ramamurthy, M. Gebauer, T. Bell, R. Sifa and C. Bauckhage, “From Imbalanced Classification to Supervised Outlier Detection Problems: Adversarially Trained Auto Encoders”, *International Conference on Artificial Neural Networks*, Springer, 2020 (cit. on pp. 11, 43, 55, 57, 67, 68, 70, 72, 76, 92, 94, 102, 105).
- [24] M. Lübbering, M. Gebauer, R. Ramamurthy, R. Sifa and C. Bauckhage, “Supervised Autoencoder Variants for End to End Anomaly Detection”, *Pattern Recognition. ICPR International Workshops and Challenges*, 2021 (cit. on pp. 11, 43, 44, 68, 72, 102, 105).

-
- [25] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota and T. E. Boulton, *Toward open set recognition*, *IEEE transactions on pattern analysis and machine intelligence* **35** (2012) (cit. on pp. 11, 20, 64, 65, 72, 73, 90).
- [26] M. Lübbering, M. Gebauer, R. Ramamurthy, C. Bauckhage and R. Sifa, “From Open Set Recognition Towards Robust Multi-class Classification”, *Artificial Neural Networks and Machine Learning – ICANN 2022*, 2022 (cit. on pp. 11, 90).
- [27] E. Hüllermeier and W. Waegeman, *Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods*, *Machine Learning* **110** (2021) (cit. on pp. 11, 90–92, 95).
- [28] K. Hornik, M. Stinchcombe and H. White, *Multilayer feedforward networks are universal approximators*, *Neural networks* **2** (1989) (cit. on p. 12).
- [29] V. Nair and G. E. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines”, *ICML’10*, 2010 (cit. on p. 12).
- [30] S. Ruder, *An overview of gradient descent optimization algorithms*, arXiv preprint arXiv:1609.04747 (2016) (cit. on pp. 13, 53).
- [31] M. D. Zeiler, *Adadelta: An Adaptive Learning Rate Method*, arXiv preprint arXiv:1212.5701 (2012) (cit. on pp. 13, 53, 57).
- [32] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization”, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015 (cit. on pp. 13, 80, 96).
- [33] D. Hendrycks and K. Gimpel, *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks*, *Proceedings of International Conference on Learning Representations* (2017) (cit. on pp. 14, 18, 22, 64, 72, 76, 81, 90, 96).
- [34] D. Hendrycks, M. Mazeika and T. Dietterich, *Deep anomaly detection with outlier exposure*, arXiv preprint arXiv:1812.04606 (2018) (cit. on pp. 14, 23, 64, 65, 81).
- [35] B. Lakshminarayanan, A. Pritzel and C. Blundell, *Simple and scalable predictive uncertainty estimation using deep ensembles*, *Advances in neural information processing systems* **30** (2017) (cit. on pp. 14, 22, 64, 68, 76, 83, 90–92, 96, 100, 103).
- [36] A. Nguyen, J. Yosinski and J. Clune, “Deep Neural Networks Are Easily Fooled: High Confidence Predictions for Unrecognizable Images”, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015 (cit. on pp. 14, 64).
- [37] I. Goodfellow, J. Shlens and C. Szegedy, “Explaining and Harnessing Adversarial Examples”, *International Conference on Learning Representations*, 2015 (cit. on pp. 14, 22, 64, 74).
- [38] F. E. Grubbs, *Procedures for detecting outlying observations in samples*, *Technometrics* **11** (1969) (cit. on p. 16).

- [39] C. Geng, S.-j. Huang and S. Chen, *Recent advances in open set recognition: A survey*, IEEE transactions on pattern analysis and machine intelligence (2020) (cit. on pp. 16, 72).
- [40] P. Perera, P. Oza and V. M. Patel, *One-class classification: A survey*, arXiv preprint arXiv:2101.03064 (2021) (cit. on p. 16).
- [41] M. A. Pimentel, D. A. Clifton, L. Clifton and L. Tarassenko, *A review of novelty detection*, Signal processing **99** (2014) (cit. on p. 16).
- [42] D. M. Hawkins, *Identification of Outliers*, vol. 11, Springer, 1980 (cit. on pp. 16, 44, 65).
- [43] C. C. Aggarwal, *Outlier analysis*, Springer, 2017 (cit. on pp. 16–19, 45, 65, 68).
- [44] J. Chen, S. Sathe, C. Aggarwal and D. Turaga, “Outlier Detection with Autoencoder Ensembles”, *Proceedings of the SIAM international conference on data mining*, 2017 (cit. on pp. 17, 46).
- [45] G. Pang, C. Shen, L. Cao and A. V. D. Hengel, *Deep learning for anomaly detection: A review*, ACM Computing Surveys (CSUR) **54** (2021) (cit. on pp. 17, 19, 46).
- [46] R. Chalapathy and S. Chawla, *Deep Learning for Anomaly Detection: A Survey*, arXiv preprint arXiv:1901.03407 (2019) (cit. on pp. 17, 19, 46).
- [47] C. C. Aggarwal, A. Hinneburg and D. A. Keim, “On the surprising behavior of distance metrics in high dimensional space”, *International conference on database theory*, 2001 (cit. on p. 17).
- [48] A. Hinneburg, C. C. Aggarwal and D. A. Keim, “What is the nearest neighbor in high dimensional spaces?”, *26th Internat. Conference on Very Large Databases*, 2000 (cit. on p. 17).
- [49] S. Wang, C. Liu, X. Gao, H. Qu and W. Xu, “Session-based fraud detection in online e-commerce transactions using recurrent neural networks”, *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2017 (cit. on p. 17).
- [50] Y. Yu, J. Long and Z. Cai, *Network intrusion detection through stacking dilated convolutional autoencoders*, Security and Communication Networks **2017** (2017) (cit. on p. 17).
- [51] C. Cao, F. Liu, H. Tan, D. Song, W. Shu, W. Li, Y. Zhou, X. Bo and Z. Xie, *Deep learning and its applications in biomedicine, Genomics, proteomics & bioinformatics* **16** (2018) (cit. on p. 17).
- [52] S. Min, B. Lee and S. Yoon, *Deep learning in bioinformatics, Briefings in bioinformatics* **18** (2017) (cit. on p. 17).
- [53] T. Fawcett, *An Introduction to ROC Analysis*, Pattern recognition letters (2006) (cit. on p. 18).
- [54] T. Saito and M. Rehmsmeier, *The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets*, PloS one **10** (2015) (cit. on p. 18).
- [55] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D.-k. Cho and H. Chen, “Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection”, *International Conference on Learning Representations*, 2018 (cit. on p. 18).

-
- [56] L. Li, R. J. Hansman, R. Palacios and R. Welsch, *Anomaly detection via a Gaussian Mixture Model for flight operation and safety monitoring*, *Transportation Research Part C: Emerging Technologies* **64** (2016) (cit. on p. 18).
- [57] E. M. Knox and R. T. Ng, “Algorithms for mining distancebased outliers in large datasets”, *Proceedings of the international conference on very large data bases*, Citeseer, 1998 (cit. on p. 18).
- [58] A. Bendale and T. E. Boult, “Towards Open Set Deep Networks”, *Proceedings of the IEEE Conference on Computer Vision and Rattern Recognition*, 2016 (cit. on pp. 18, 67, 72, 73, 75, 90, 92).
- [59] J. M. Johnson and T. M. Khoshgoftaar, *Survey on deep learning with class imbalance*, *Journal of Big Data* **6** (2019) (cit. on pp. 19, 46).
- [60] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, *SMOTE: synthetic minority over-sampling technique*, *Journal of artificial intelligence research* **16** (2002) (cit. on pp. 19, 44, 46).
- [61] B. Krawczyk, *Learning from imbalanced data: open challenges and future directions*, *Progress in Artificial Intelligence* **5** (2016) (cit. on p. 19).
- [62] J. Andrews, T. Tanay, E. J. Morton and L. D. Griffin, “Transfer representation-learning for anomaly detection”, *JMLR*, 2016 (cit. on p. 19).
- [63] D. Xu, E. Ricci, Y. Yan, J. Song and N. Sebe, *Learning deep representations of appearance and motion for anomalous event detection*, arXiv preprint arXiv:1510.01553 (2015) (cit. on pp. 19, 68).
- [64] S. Hawkins, H. He, G. Williams and R. Baxter, “Outlier Detection using Replicator Neural Networks”, *Proceedings of International Conference on Data Warehousing and Knowledge Discovery*, 2002 (cit. on pp. 19, 45, 46, 68, 102).
- [65] C. Zhou and R. C. Paffenroth, “Anomaly detection with robust deep autoencoders”, *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017 (cit. on pp. 19, 46).
- [66] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth and G. Langs, “Unsupervised Anomaly Detection with Generative Adversarial Networks to guide Marker Discovery”, *Proceedings International Conference on Information Processing in Medical Imaging*, 2017 (cit. on pp. 19, 65, 68).
- [67] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller and M. Kloft, “Deep Semi-Supervised Anomaly Detection”, *International Conference on Learning Representations*, 2020 (cit. on pp. 19, 102).
- [68] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller and M. Kloft, “Deep One-Class Classification”, *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, PMLR, 2018 (cit. on pp. 19, 20, 65, 68).
- [69] C. M. Bishop, *Pattern recognition and machine learning*, Springer, 2006 (cit. on pp. 20, 64, 67).

- [70] M. M. Moya, M. W. Koch and L. D. Hostetler, *One-class classifier networks for target recognition applications*, NASA STI/Recon Technical Report N 93 (1993) (cit. on pp. 20, 65).
- [71] J. M. Lee, *Introduction to smooth manifolds*, Springer, 2013 (cit. on p. 20).
- [72] S. Tong, H. Gu and K. Yu, “A comparative study of robustness of deep learning approaches for VAD”, *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2016 (cit. on p. 22).
- [73] N. Drenkow, N. Sani, I. Shpitser and M. Unberath, *Robustness in Deep Learning for Computer Vision: Mind the gap?*, arXiv preprint arXiv:2112.00639 (2021) (cit. on p. 22).
- [74] F. Yu, Z. Qin, C. Liu, L. Zhao, Y. Wang and X. Chen, *Interpreting and evaluating neural network robustness*, arXiv preprint arXiv:1905.04270 (2019) (cit. on pp. 22, 75).
- [75] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay and D. Mukhopadhyay, *Adversarial attacks and defences: A survey*, arXiv preprint arXiv:1810.00069 (2018) (cit. on p. 22).
- [76] N. Akhtar and A. Mian, *Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey*, *IEEE Access* 6 (2018) (cit. on p. 22).
- [77] C. Guo, G. Pleiss, Y. Sun and K. Q. Weinberger, “On Calibration of Modern Neural Networks”, *International Conference on Machine Learning*, 2017 (cit. on pp. 22, 68, 76, 85, 90).
- [78] M. P. Naeni, G. Cooper and M. Hauskrecht, “Obtaining Well Calibrated Probabilities Using Bayesian Binning”, *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015 (cit. on pp. 22, 76).
- [79] M. Kull, M. Perello Nieto, M. Kängsepp, T. Silva Filho, H. Song and P. Flach, “Beyond Temperature Scaling: Obtaining Well-calibrated Multi-class Probabilities with Dirichlet Calibration”, *Advances in Neural Information Processing Systems*, 2019 (cit. on pp. 22, 76).
- [80] S. Liang, Y. Li and R. Srikant, *Enhancing the reliability of out-of-distribution image detection in neural networks*, arXiv preprint arXiv:1706.02690 (2017) (cit. on p. 23).
- [81] M. Lübbering, M. Pielka, I. Henk and R. Sifa, “Datastack: unification of heterogeneous machine learning dataset interfaces”, *2022 IEEE 38th International Conference on Data Engineering Workshops (ICDEW)*, IEEE, 2022 66 (cit. on p. 26).
- [82] R. Tatman, J. VanderPlas and S. Dane, *A practical taxonomy of reproducibility for machine learning research*, (2018) (cit. on pp. 26, 31).

-
- [83] J. Forde, T. Head, C. Holdgraf, Y. Panda, G. Nalvarete, B. Ragan-Kelley and E. Sundell, *Reproducible research environments with repo2docker*, (2018) (cit. on pp. 26, 31).
- [84] O. E. Gundersen and S. Kjensmo, “State of the art: Reproducibility in artificial intelligence”, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 1, 2018 (cit. on pp. 26, 31, 33).
- [85] M. Hutson, *Artificial intelligence faces reproducibility crisis*, *Science* **359** (2018) (cit. on pp. 26, 31).
- [86] W. McKinney et al., “Data structures for statistical computing in python”, *Proceedings of the 9th Python in Science Conference*, vol. 445, 2010 (cit. on p. 26).
- [87] C. R. Harris et al., *Array programming with NumPy*, *Nature* **585** (2020) (cit. on pp. 26, 114).
- [88] E. Freeman, E. Robson, B. Bates and K. Sierra, *Head first design patterns*, " O'Reilly Media, Inc.", 2008 (cit. on p. 27).
- [89] E. Gamma, *Design patterns: elements of reusable object-oriented software*, Pearson Education India, 1995 (cit. on p. 27).
- [90] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal and J. Gama, *Machine learning for streaming data: state of the art, challenges, and opportunities*, *ACM SIGKDD Explorations Newsletter* **21** (2019) (cit. on p. 27).
- [91] S. Landset, T. M. Khoshgoftaar, A. N. Richter and T. Hasanin, *A survey of open source tools for machine learning with big data in the Hadoop ecosystem*, *Journal of Big Data* **2** (2015) (cit. on p. 27).
- [92] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi and K. Tzoumas, *Apache flink: Stream and batch processing in a single engine*, *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* **36** (2015) (cit. on p. 28).
- [93] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham et al., “Storm@ twitter”, *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014 (cit. on p. 28).
- [94] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker and I. Stoica, “Discretized streams: Fault-tolerant streaming computation at scale”, *Proceedings of the twenty-fourth ACM symposium on operating systems principles*, 2013 (cit. on p. 28).
- [95] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al., *Scikit-learn: Machine learning in Python*, *Journal of Machine Learning Research* **12** (2011) (cit. on p. 28).
- [96] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin et al., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*, arXiv preprint arXiv:1603.04467 (2016) (cit. on pp. 28, 32, 33).

- [97] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimeshain, L. Antiga et al.,
Pytorch: An imperative style, high-performance deep learning library,
Advances in neural information processing systems **32** (2019) (cit. on pp. 28, 32, 33).
- [98] S. Kaufman, S. Rosset and C. Perlich,
“Leakage in Data Mining: Formulation, Detection, and Avoidance”, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*,
KDD ’11, Association for Computing Machinery, 2011 (cit. on p. 28).
- [99] M. Lübbering, M. Gebauer, R. Ramamurthy, M. Pielka, C. Bauckhage and R. Sifa,
“Utilizing representation learning for robust text classification under datasetshift”,
Proceedings of the Conference “Lernen, Wissen, Daten, Analysen”, CEUR Workshop Proceedings, 2021 (cit. on pp. 30, 63, 76, 78, 80, 81, 95).
- [100] R. Sifa and C. Bauckhage, “Novelty Discovery with Kernel Minimum Enclosing Balls”,
International Conference on Learning and Intelligent Optimization, Springer, 2020
(cit. on p. 30).
- [101] M. Lübbering and I. Henk, *OutlierHub, a Curated Hub for Outlier and Anomaly Datasets*,
version 0.0.51, <https://github.com/fraunhofer-iais/outlierhub>, 2021 (cit. on p. 31).
- [102] M. Baker, *Reproducibility crisis*, *Nature* **533** (2016) (cit. on p. 31).
- [103] J. Pineau, K. Sinha, G. Fried, R. N. Ke and H. Larochelle,
ICLR Reproducibility Challenge 2019, *ReScience C* **5** (2019),
URL: <https://zenodo.org/record/3158244/files/article.pdf> (cit. on p. 31).
- [104] J. Pineau, P. Vincent-Lamarre, K. Sinha, V. Lariviere, A. Beygelzimer, F. d’Alche-Buc, E. Fox
and H. Larochelle, *Improving Reproducibility in Machine Learning Research(A Report from
the NeurIPS 2019 Reproducibility Program)*,
Journal of Machine Learning Research **22** (2021),
URL: <http://jmlr.org/papers/v22/20-303.html> (cit. on p. 31).
- [105] J. Dodge, S. Gururangan, D. Card, R. Schwartz and N. A. Smith,
Show Your Work: Improved Reporting of Experimental Results,
arXiv preprint arXiv:1909.03004 (2019) (cit. on pp. 31, 41).
- [106] B. K. Olorisade, P. Brereton and P. Andras,
Reproducibility in Machine Learning-Based Studies: An example of Text Mining, (2017)
(cit. on p. 31).
- [107] A. L. Beam, A. K. Manrai and M. Ghassemi,
Challenges to the reproducibility of machine learning models in health care, *Jama* **323** (2020)
(cit. on p. 31).
- [108] M. McDermott, S. Wang, N. Marinsek, R. Ranganath, M. Ghassemi and L. Foschini,
Reproducibility in Machine Learning for Health, arXiv preprint arXiv:1907.01463 (2019)
(cit. on p. 31).
- [109] E. Raff, *A step toward quantifying independently reproducible machine learning research*,
Advances in Neural Information Processing Systems **32** (2019) (cit. on p. 31).

-
- [110] D. G. Feitelson, *From repeatability to reproducibility and corroboration*, ACM SIGOPS Operating Systems Review **49** (2015) (cit. on p. 31).
- [111] G. K. Sandve, A. Nekrutenko, J. Taylor and E. Hovig, *Ten Simple Rules for Reproducible Computational Research*, PLoS Computational Biology (2013) (cit. on p. 32).
- [112] M. Mattsson, *Object-oriented frameworks*, Licentiate thesis (1996) (cit. on p. 32).
- [113] M. Fowler, *Event Sourcing*, <https://martinfowler.com/eaaDev/EventSourcing.html>, Accessed: 2022-10-06, 2005 (cit. on pp. 32, 37).
- [114] R. Schwartz, J. Dodge, N. A. Smith and O. Etzioni, *Green ai*, Communications of the ACM **63** (2020) (cit. on pp. 33, 40).
- [115] L. Biewald, *Experiment Tracking with Weights and Biases*, Software available from wandb.com, 2020, URL: <https://www.wandb.com/> (cit. on pp. 33, 34).
- [116] K. Greff, A. Klein, M. Chovanec, F. Hutter and J. Schmidhuber, “The sacred infrastructure for computational research”, *Proceedings of the 16th python in science conference*, vol. 28, 2017 (cit. on p. 33).
- [117] W. Falcon et al., *Pytorch lightning*, GitHub. Note: <https://github.com/PyTorchLightning/pytorch-lightning> **3** (2019) (cit. on p. 33).
- [118] T. Akiba, S. Sano, T. Yanase, T. Ohta and M. Koyama, “Optuna: A Next-generation Hyperparameter Optimization Framework”, *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019 (cit. on p. 34).
- [119] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez and I. Stoica, *Tune: A Research Platform for Distributed Model Selection and Training*, arXiv preprint arXiv:1807.05118 (2018) (cit. on p. 34).
- [120] A. Chen, A. Chow, A. Davidson, A. DCunha, A. Ghodsi, S. A. Hong, A. Konwinski, C. Mewald, S. Murching, T. Nykodym et al., “Developments in mlflow: A system to accelerate the machine learning lifecycle”, *Proceedings of the fourth international workshop on data management for end-to-end machine learning*, 2020 (cit. on p. 34).
- [121] S. Raschka, *Model evaluation, model selection, and algorithm selection in machine learning*, arXiv preprint arXiv:1811.12808 (2018) (cit. on pp. 35, 40, 75).
- [122] M. Fowler, *Patterns of Enterprise Application Architecture*, Addison-Wesley, 2002 (cit. on p. 36).
- [123] C. Larman, *Protected variation: The importance of being closed*, IEEE Software **18** (2001) (cit. on p. 36).

- [124] J. Rasley, S. Rajbhandari, O. Ruwase and Y. He, “Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters”, *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020 (cit. on p. 40).
- [125] J. Cowls, A. Tsamados, M. Taddeo and L. Floridi, *The AI gambit: leveraging artificial intelligence to combat climate change—opportunities, challenges, and recommendations*, *Ai & Society* (2021) (cit. on p. 40).
- [126] E. Strubell, A. Ganesh and A. McCallum, *Energy and policy considerations for deep learning in NLP*, arXiv preprint arXiv:1906.02243 (2019) (cit. on p. 41).
- [127] N. Japkowicz and S. Stephen, *The class imbalance problem: A systematic study*, *Intelligent Data Analysis* **6** (2002) (cit. on p. 44).
- [128] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker and G. D. Tourassi, *Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance*, *Neural networks* (2008) (cit. on p. 44).
- [129] D. Olszewski, *A probabilistic approach to fraud detection in telecommunications*, *Knowledge-Based Systems* **26** (2012) (cit. on p. 44).
- [130] S. Panigrahi, A. Kundu, S. Sural and A. K. Majumdar, *Credit card fraud detection: A fusion approach using Dempster–Shafer theory and Bayesian learning*, *Information Fusion* **10** (2009) (cit. on p. 44).
- [131] M. Kubat, R. C. Holte and S. Matwin, *Machine learning for the detection of oil spills in satellite radar images*, *Machine learning* **30** (1998) (cit. on p. 44).
- [132] C. Huang, Y. Li, C. Change Loy and X. Tang, “Learning deep representation for imbalanced classification”, *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2016 (cit. on p. 44).
- [133] M. Buda, A. Maki and M. A. Mazurowski, *A Systematic Study of the Class Imbalance Problem in CNNs*, *Neural Networks* (2018) (cit. on p. 44).
- [134] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman and A. Galstyan, *A survey on Bias and Fairness in Machine Learning*, arXiv preprint arXiv:1908.09635 (2019) (cit. on p. 44).
- [135] C. Cardie and N. Howe, “Improving Minority Class Prediction using Case-specific Feature Weights”, *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997 (cit. on p. 44).
- [136] S. Lawrence, I. Burns, A. Back, A. C. Tsoi and C. L. Giles, “Neural Network Classification and Prior Class Probabilities”, *Neural Networks: Tricks of the Trade*, 1998 (cit. on p. 44).
- [137] M. Kukar, I. Kononenko et al., “Cost-sensitive Learning with Neural Networks.”, *Proceedings of European Conference on Artificial Intelligence*, 1998 (cit. on p. 44).

-
- [138] N. Japkowicz, C. Myers, M. Gluck et al., “A Novelty Detection Approach to Classification”, *Proceedings of International Joint Conference on Artificial Intelligence*, 1995 (cit. on p. 44).
- [139] H.-j. Lee and S. Cho,
“The Novelty Detection Approach for Different degrees of Class Imbalance”,
Proceedings of International Conference on Neural Information processing, 2006
(cit. on p. 44).
- [140] B. Scholkopf and A. J. Smola,
Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond,
MIT Press, 2001 (cit. on p. 44).
- [141] Y. Ishii and M. Takanashi,
Low-cost Unsupervised Outlier Detection by Autoencoders with Robust Estimation,
Journal of Information Processing (2019) (cit. on pp. 44, 46, 47).
- [142] H. Dutta, C. Giannella, K. Borne and H. Kargupta,
“Distributed top-k Outlier Detection from Astronomy Catalogs using the Demac System”,
Proceedings of the 2007 SIAM International Conference on Data Mining, 2007 (cit. on p. 44).
- [143] L. J. Latecki, A. Lazarevic and D. Pokrajac,
“Outlier Detection with Kernel Density Functions”, *In Proceedings of International Workshop on Machine Learning and Data Mining in Pattern Recognition*, 2007 (cit. on p. 44).
- [144] Y. S. Chong and Y. H. Tay,
“Abnormal Event Detection in Videos using Spatiotemporal Autoencoder”,
Proceedings of International Symposium on Neural Nets. 2017 (cit. on pp. 44, 45).
- [145] J. Zou, J. Zhang and P. Jiang,
Credit Card Fraud Detection Using Autoencoder Neural Network,
arXiv preprint arXiv:1908.11553 (2019) (cit. on pp. 44, 65).
- [146] V. Chandola, A. Banerjee and V. Kumar, *Anomaly Detection: A Survey*,
ACM Comput. Surv. **41** (2009) (cit. on p. 45).
- [147] H. A. Dau, V. Ciesielski and A. Song,
“Anomaly Detection Using Replicator Neural Networks Trained on Examples of One Class”,
Proceedings of the 10th International Conference on Simulated Evolution and Learning, 2014
(cit. on pp. 45, 46, 68).
- [148] R. Chalapathy, A. K. Menon and S. Chawla,
Anomaly Detection using One-Class Neural Networks,
arXiv preprint arXiv:1802.06360 (2018) (cit. on pp. 45, 46, 68).
- [149] J. An and S. Cho,
Variational Autoencoder based Anomaly Detection using Reconstruction Probability,
Special Lecture on IE (2015) (cit. on p. 45).
- [150] H. Sarvari, C. Domeniconi, B. Prencakj and G. Stilo,
Unsupervised Boosting-based Autoencoder Ensembles for Outlier Detection,
arXiv preprint arXiv:1910.09754 (2019) (cit. on p. 46).

- [151] J. Van Hulse, T. M. Khoshgoftaar and A. Napolitano, “Experimental perspectives on learning from imbalanced data”, *Proceedings of the 24th international conference on Machine learning*, 2007 (cit. on p. 46).
- [152] H. Wang, Z. Cui, Y. Chen, M. Avidan, A. B. Abdallah and A. Kronzer, *Predicting hospital readmission via cost-sensitive deep learning*, *IEEE/ACM transactions on computational biology and bioinformatics* **15** (2018) (cit. on p. 46).
- [153] C. Zhang, K. C. Tan and R. Ren, “Training cost-sensitive deep belief networks on imbalance data problems”, *2016 international joint conference on neural networks (IJCNN)*, IEEE, 2016 (cit. on p. 46).
- [154] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, “Focal loss for dense object detection”, *Proceedings of the IEEE international conference on computer vision*, 2017 (cit. on p. 46).
- [155] W. Ng, G. Zeng, J. Zhang, D. Yeung and W. Pedrycz, *Dual Autoencoders Features for Imbalance Classification Problem*, *Pattern Recognition* (2016) (cit. on p. 46).
- [156] J. Pennington, R. Socher and C. D. Manning, “Glove: Global Vectors for Word Representation”, *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014 (cit. on pp. 46, 78, 95).
- [157] Y. Yang and X. Liu, “A re-examination of Text Categorization Methods”, *Proceedings of Int Conference on Research and Development in Information Retrieval*, 1999 (cit. on p. 46).
- [158] T. Joachims, “Text Categorization with Support Vector Machines: Learning with Many Relevant Features”, *Proceedings of European Conference on Machine Learning*, 1998 (cit. on p. 46).
- [159] R. Kannan, H. Woo, C. C. Aggarwal and H. Park, “Outlier Detection for Text Data”, *Proceedings of International Conference on Data Mining*, 2017 (cit. on p. 46).
- [160] V. Hautamaki, I. Karkkainen and P. Franti, “Outlier Detection using k-Nearest Neighbour Graph”, *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 3, 2004 (cit. on p. 47).
- [161] A. Divekar, M. Parekh, V. Savla, R. Mishra and M. Shirole, “Benchmarking Datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives”, *Proceedings of 3rd International Conference on Computing, Communication and Security (ICCCS)*, 2018 (cit. on p. 47).
- [162] P. Gogoi, B. Borah, D. Bhattacharyya and J. Kalita, *Outlier Identification using Symmetric Neighborhoods*, *Procedia Technology* **6** (2012) (cit. on p. 47).
- [163] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, “A detailed Analysis of the KDD CUP 99 Data Set”, *Proceedings of IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009 (cit. on p. 47).

-
- [164] L. McInnes, J. Healy and J. Melville, *Umap: Uniform Manifold Approximation and Projection for Dimension Reduction*, arXiv preprint arXiv:1802.03426 (2018) (cit. on pp. 48–50, 58).
- [165] L. Beggel, M. Pfeiffer and B. Bischl, “Robust anomaly detection in images using adversarial autoencoders”, *Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019 (cit. on p. 48).
- [166] B. Neyshabur, S. Bhojanapalli, D. McAllester and N. Srebro, “Exploring Generalization in Deep Learning”, *Advances in Neural Information Processing Systems*, 2017 (cit. on p. 54).
- [167] K. Kawaguchi, L. P. Kaelbling and Y. Bengio, *Generalization in Deep Learning*, arXiv preprint arXiv:1710.05468 (2017) (cit. on p. 54).
- [168] R. Caruana, *Multitask learning*, Machine learning (1997) (cit. on p. 54).
- [169] M. Ranzato and M. Szummer, “Semi-supervised Learning of Compact Document Representations with Deep Networks”, *Proceedings of International Conference on Machine learning*, 2008 (cit. on p. 54).
- [170] J. Weston, F. Ratle, H. Mobahi and R. Collobert, “Deep Learning via Semi-supervised Embedding”, *Neural networks: Tricks of the trade*, 2012 (cit. on p. 54).
- [171] J. Davis and M. Goadrich, “The relationship between Precision-Recall and ROC curves”, *Proceedings of the 23rd international conference on Machine learning*, 2006 (cit. on pp. 57, 76, 104).
- [172] K. Boyd, K. H. Eng and C. D. Page, “Area under the precision-recall curve: point estimates and confidence intervals”, *Joint European conference on machine learning and knowledge discovery in databases*, Springer, 2013 (cit. on p. 57).
- [173] M. Lübbering, M. Gebauer, R. Ramamurthy, C. Bauckhage and R. Sifa, *Bounding open space risk with decoupling autoencoders in open set recognition*, *International Journal of Data Science and Analytics* (2022) 1 (cit. on p. 63).
- [174] M. Havasi, R. Jenatton, S. Fort, J. Z. Liu, J. Snoek, B. Lakshminarayanan, A. M. Dai and D. Tran, *Training Independent Subnetworks for Robust Prediction*, arXiv preprint (2020) (cit. on pp. 64, 66–68, 76, 83, 90–92, 96, 100, 103).
- [175] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov and Q. V. Le, *Xlnet: Generalized autoregressive pretraining for language understanding*, *Advances in neural information processing systems* **32** (2019) (cit. on p. 64).
- [176] K. He, X. Zhang, S. Ren and J. Sun, “Deep residual learning for image recognition”, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016 (cit. on p. 64).
- [177] Y. LeCun, Y. Bengio and G. Hinton, *Deep learning*, *nature* (2015) (cit. on p. 64).

- [178] A. Paullada, I. D. Raji, E. M. Bender, E. Denton and A. Hanna, *Data and its (dis) contents: A survey of dataset development and use in machine learning research*, *Patterns* **2** (2021) (cit. on p. 64).
- [179] T. E. Boulton, S. Cruz, A. R. Dhamija, M. Gunther, J. Henrydoss and W. J. Scheirer, “Learning and the Unknown: Surveying Steps Toward Open World Recognition”, *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019 (cit. on pp. 64, 67, 68, 72, 90).
- [180] A. Linden and J. Kindermann, “Inversion of multilayer nets”, *Proceedings International Joint Conference Neural Networks*, vol. 2, 1989 (cit. on pp. 65, 76).
- [181] R. Girshick, “Fast r-cnn”, *Proceedings of the IEEE international conference on computer vision*, 2015 (cit. on pp. 65, 67, 76).
- [182] S. Ren, K. He, R. Girshick and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, *Advances in neural information processing systems* **28** (2015) (cit. on pp. 65, 67, 76, 110).
- [183] C. C. Aggarwal and S. Sathe, *Outlier Ensembles: An Introduction*, Springer, 2017 (cit. on p. 65).
- [184] X. Chen and E. Konukoglu, *Unsupervised Detection of Lesions in Brain MRI using constrained Adversarial Auto-Encoders*, arXiv preprint arXiv:1806.04972 (2018) (cit. on p. 65).
- [185] N. Wang, C. Chen, Y. Xie and L. Ma, *Brain tumor anomaly detection via latent regularized adversarial network*, arXiv preprint arXiv:2007.04734 (2020) (cit. on p. 65).
- [186] R. Domingues, *Probabilistic Modeling for Novelty Detection with Applications to Fraud Identification*, arXiv preprint arXiv:1903.01730 (2019) (cit. on p. 65).
- [187] C. Phua, V. Lee, K. Smith and R. Gayler, *A comprehensive survey of data mining-based fraud detection research*, arXiv preprint arXiv:1009.6119 (2010) (cit. on p. 65).
- [188] M. Gharib, B. Mohammadi, S. H. Dastgerdi and M. Sabokrou, *AutoIDS: Auto-encoder based Method for Intrusion Detection System*, arXiv preprint arXiv:1911.03306 (2019) (cit. on p. 65).
- [189] O. Kaynar, A. G. Yüksek, Y. Görmez and Y. E. Işık, “Intrusion Detection with Autoencoder based Deep Learning Machine”, *Proceedings of 25th Signal Processing and Communications Applications Conference (SIU)*, 2017 (cit. on p. 65).
- [190] P. Oza and V. M. Patel, “C2ae: Class conditioned auto-encoder for open-set recognition”, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019 (cit. on pp. 67, 72, 75).

-
- [191] D. H. Fusilier, R. G. Cabrera, M. Montes and P. Rosso, “Using PU-learning to detect deceptive opinion spam”, *Proceedings of the 4th workshop on computational approaches to subjectivity, sentiment and social media analysis*, 2013 (cit. on p. 67).
- [192] U. Tanielian and F. Vasile, “Relaxed softmax for PU learning”, *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019 (cit. on p. 67).
- [193] C.-J. Hsieh, N. Natarajan and I. Dhillon, “PU learning for matrix completion”, *International Conference on Machine Learning*, PMLR, 2015 (cit. on p. 67).
- [194] E. M. Rudd, L. P. Jain, W. J. Scheirer and T. E. Boult, *The extreme value machine*, *IEEE transactions on pattern analysis and machine intelligence* **40** (2017) (cit. on p. 67).
- [195] A. Rozsa and T. E. Boult, *Improved adversarial robustness by reducing open space risk via tent activations*, arXiv preprint arXiv:1908.02435 (2019) (cit. on pp. 67, 73).
- [196] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor and J. C. Platt, “Support Vector Method for Novelty Detection”, *In Proceedings of Advances in Neural Information Processing Systems*, 2000 (cit. on p. 68).
- [197] S. M. Erfani, S. Rajasegarar, S. Karunasekera and C. Leckie, *High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning*, *Pattern Recognition* **58** (2016) (cit. on p. 68).
- [198] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, *Generative adversarial networks*, arXiv preprint arXiv:1406.2661 (2014) (cit. on p. 68).
- [199] D. P. Kingma and M. Welling, “Auto-encoding variational bayes”, 2014 (cit. on p. 68).
- [200] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng et al., “Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications”, *Proceedings of the 2018 World Wide Web Conference*, 2018 (cit. on p. 68).
- [201] S. Nedelkoski, J. Cardoso and O. Kao, “Anomaly detection and classification using distributed tracing and deep learning”, *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2019 (cit. on p. 68).
- [202] M. Lübbering, M. Pielka, K. Das, M. Gebauer, R. Ramamurthy, C. Bauckhage and R. Sifa, “Toxicity Detection in Online Comments with Limited Data: A Comparative Analysis.”, *ESANN*, 2021 (cit. on pp. 68, 102).
- [203] T. DeVries and G. W. Taylor, *Learning confidence for out-of-distribution detection in neural networks*, arXiv preprint arXiv:1802.04865 (2018) (cit. on p. 68).
- [204] J. Van Amersfoort, L. Smith, Y. W. Teh and Y. Gal, “Uncertainty Estimation Using a Single Deep Deterministic Neural Network”, *International Conference on Machine Learning*, PMLR, 2020 (cit. on pp. 68, 72, 75, 82, 90–92, 96, 100, 120).

- [205] S. Varma and R. Simon, *Bias in Error Estimation when Using Cross-validation for Model Selection*, BMC bioinformatics **7** (2006) (cit. on p. 75).
- [206] G. W. Brier, *Verification of Forecasts Expressed in Terms of Probability*, Monthly Weather Review **78** (1950) (cit. on p. 76).
- [207] X. Li and D. Roth, “Learning Question Classifiers”, *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002 (cit. on pp. 77, 106).
- [208] Y. LeCun, *The MNIST database of handwritten digits*, <http://yann.lecun.com/exdb/mnist/> (1998) (cit. on p. 78).
- [209] H. Xiao, K. Rasul and R. Vollgraf, *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*, arXiv preprint arXiv:1708.07747 (2017) (cit. on p. 78).
- [210] R. G. Lopes, D. Yin, B. Poole, J. Gilmer and E. D. Cubuk, *Improving robustness without sacrificing accuracy with patch gaussian augmentation*, arXiv preprint arXiv:1906.02611 (2019) (cit. on p. 83).
- [211] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer and B. Lakshminarayanan, *Augmix: A simple data processing method to improve robustness and uncertainty*, arXiv preprint arXiv:1912.02781 (2019) (cit. on p. 83).
- [212] B. Zadrozny and C. Elkan, “Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers”, *Proceedings of the Eighteenth International Conference on Machine Learning*, vol. 1, ICML ’01, Citeseer, 2001 (cit. on p. 85).
- [213] D. Feng, L. Rosenbaum and K. Dietmayer, “Towards Safe Autonomous Driving: Capture Uncertainty in the Deep Neural Network for Lidar 3d Vehicle Detection”, *21st International Conference on Intelligent Transportation Systems*, IEEE, 2018 (cit. on p. 90).
- [214] A. Lambrou, H. Papadopoulos and A. Gammerman, *Reliable Confidence Measures for Medical Diagnosis With Evolutionary Algorithms*, IEEE Transactions on Information Technology in Biomedicine **15** (2010) (cit. on p. 90).
- [215] A. Kendall and Y. Gal, *What uncertainties do we need in bayesian deep learning for computer vision?*, Advances in neural information processing systems **30** (2017) (cit. on pp. 90, 92).
- [216] Y. Gal, *Uncertainty in Deep Learning*, PhD thesis: University of Cambridge, 2016 (cit. on pp. 90–92).
- [217] D. J. MacKay, *A Practical Bayesian Framework for Backpropagation Networks*, Neural computation **4** (1992) (cit. on p. 92).
- [218] S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez and S. Udfluft, “Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-Sensitive Learning”, *International Conference on Machine Learning*, 2018 (cit. on p. 92).

-
- [219] A. Mobiny, P. Yuan, S. K. Moulik, N. Garg, C. C. Wu and H. Van Nguyen, *Dropconnect Is Effective in Modeling Uncertainty of Bayesian Deep Networks*, *Scientific reports* **11** (2021) (cit. on p. 92).
- [220] J. van Amersfoort, L. Smith, A. Jesson, O. Key and Y. Gal, *On feature collapse and deep kernel learning for single forward pass uncertainty*, arXiv preprint arXiv:2102.11409 (2021) (cit. on pp. 92, 120).
- [221] J. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax Weiss and B. Lakshminarayanan, *Simple and Principled Uncertainty Estimation With Deterministic Deep Learning via Distance Awareness*, *Advances in Neural Information Processing Systems* **33** (2020) (cit. on pp. 92, 120).
- [222] R. Rajkumar, M. Lübbering, T. Bell, M. Gebauer, B. Ulusay, D. Uedelhoven, T. Dilmaghani, R. Loitz, M. Pielka, C. Bauckhage and R. Sifa, “Automatic Indexing of Financial Documents via Information Extraction”, *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2021 (cit. on p. 102).
- [223] M. Duggan, *Online Harassment 2017*, Pew Research Center (2017) (cit. on p. 102).
- [224] R. K. Betty van Aken Julian Risch and A. Löser, *Challenges for Toxic Comment Classification: An In-Depth Error Analysis*, EMNLP (2018) (cit. on p. 102).
- [225] J. AI, *Toxic Comment Classification Challenge*, 1999, URL: <https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data> (visited on) (cit. on p. 103).
- [226] R. Agombar, M. Luebbering and R. Sifa, “A Clustering Backed Deep Learning Approach for Document Layout Analysis”, *Proceedings of International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, 2020 (cit. on pp. 108, 110).
- [227] R. Sifa, A. Ladi, M. Pielka, R. Ramamurthy, L. Hillebrand, B. Kirsch, D. Biesner, R. Stenzel, T. Bell, M. Lübbering, U. Nütten, C. Bauckhage, U. Warning, B. Fürst, T. Dilmaghani Khameneh, D. Thom, I. Huseynov, J. Kahlert R. amd Schlums, H. Ismail, B. Kliem and R. Loitz, “Towards automated auditing with machine learning”, *Proceedings of the ACM Symposium on Document Engineering 2019*, 2019 (cit. on pp. 108, 109).
- [228] E. Brito, R. Sifa, C. Bauckhage, R. Loitz, U. Lohmeier and C. Pünt, “A Hybrid AI Tool to Extract Key Performance Indicators from Financial Reports for Benchmarking”, *Proceedings of DocEng*, 2019 (cit. on p. 108).
- [229] T. Goel, P. Jain, I. Verma, L. Dey and S. Paliwal, “Mining company sustainability reports to aid financial decision-making”, *Proceedings of AAAI Workshop on Know. Disc. from Unstructured Data in Fin. Services*, 2020 (cit. on p. 109).
- [230] J. C. Salinas Alvarado, K. Verspoor and T. Baldwin, “Domain Adaption of Named Entity Recognition to Support Credit Risk Assessment”, *Proceedings of Australasian Language Technology Association Workshop*, 2015 (cit. on p. 109).

- [231] D. Biesner, R. Ramamurthy, R. Stenzel, M. Lübbering, L. Hillebrand, A. Ladi, M. Pielka, R. Loitz, C. Bauckhage and R. Sifa, *Anonymization of German financial documents using neural network-based language models with contextual word representations*, *International Journal of Data Science and Analytics* (2021) (cit. on p. 109).
- [232] M. Bulla, L. Hillebrand, M. Lübbering and R. Sifa, “Knowledge Graph Based Question Answering System for Financial Securities”, *German Conference on Artificial Intelligence (Künstliche Intelligenz)*, Springer, 2021 (cit. on p. 109).
- [233] P. Gupta, S. Rajaram, H. Sch"utze and T. Runkler, “Neural relation extraction within and across sentence boundaries”, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 01, 2019 (cit. on p. 109).
- [234] X. Wang, Y. Jiang, N. Bach, T. Wang, Z. Huang, F. Huang and K. Tu, *Automated Concatenation of Embeddings for Structured Prediction*, arXiv preprint arXiv:2010.05006 (2020) (cit. on p. 109).
- [235] S. Edunov, M. Ott, M. Auli and D. Grangier, *Understanding back-translation at scale*, arXiv preprint arXiv:1808.09381 (2018) (cit. on p. 109).
- [236] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, *Operating systems: Three easy pieces*, Arpaci-Dusseau Books LLC, 2018 (cit. on p. 109).
- [237] B. Grand, *Patterns in Java, Volume 2, A Catalog of Reusable Design Patterns*, 1999 (cit. on p. 109).
- [238] M. Lui and T. Baldwin, “langid.py: An off-the-shelf language identification tool”, *Proceedings of the ACL 2012 system demonstrations*, 2012 (cit. on p. 111).
- [239] A. Akbik, D. Blythe and R. Vollgraf, “Contextual String Embeddings for Sequence Labeling”, *Proceedings of International Conference on Computational Linguistics*, 2018 (cit. on p. 111).
- [240] P. Singh, S. Varadarajan, A. N. Singh and M. M. Srivastava, “Multi-domain Document Layout Understanding Using Few-Shot Object Detection”, *Proceedings of International Conference Image Analysis and Recognition*, 2020 (cit. on p. 110).
- [241] S. Schreiber et al., “Deepdesrt: Deep Learning for Detection and Structure Recognition of Tables in Document Images”, *Proceedings of International Conference on Document Analysis and Recognition*, 2017 (cit. on p. 110).
- [242] C. Soto and S. Yoo, “Visual Detection with Context for Document Layout Analysis”, *Proceedings of Conference on EMNLP-IJCNLP*, 2019 (cit. on p. 110).
- [243] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”, *Proceedings of KDD*, 1996 (cit. on p. 110).
- [244] R. Smith, “An Overview of the Tesseract OCR Engine”, *Proceedings of International Conference on Doc. Analysis and Recognition*, 2007 (cit. on p. 110).

-
- [245] T. Miyato, T. Kataoka, M. Koyama and Y. Yoshida,
Spectral normalization for generative adversarial networks,
arXiv preprint arXiv:1802.05957 (2018) (cit. on p. 120).

List of Figures

- 1.1 Exemplary architecture of a feedforward neural network: Each neuron $\varphi_i^{(k)}$ in layer k calculates the weighted sum of the output in the preceding layer and the weighted $b = 1$ and subsequently maps it onto a scalar value via a non-linear activation function $a : \mathbb{R} \rightarrow \mathbb{R}$. The final output layer o predicts a sample as one of the classes according to the maximum output value. These outputs are also referred to as logits and are usually mapped to probability scores via the sigmoid function in the case of binary classification or the softmax function in case of multi-class classification. The objective during DNN training is to adapt the weights such that the true target function $f^* : \mathbb{X} \rightarrow \mathbb{Y}$ is approximated accurately. 12

- 1.2 Demonstrating the intrinsic issues of empirical risk minimization in DNNs: We trained an MLP on each binary toy dataset (red and blue samples) and highlighted the decision boundary surface. While the model generalizes well within the classes it was exposed to at training time, each noise sample (orange) is attributed to one of the two classes with high confidence, jeopardizing model robustness. We would expect a robust model to learn a hull around the inlier data (i.e., red and blue samples), efficiently rejecting any outlier data at test time. We will propose such methods based on reconstructive representation learning in this thesis. 15

- 1.3 Different perspectives on the outlieriness of a sample: Training set D_1 contains only a single inlier class, training set D_2 contains three disjoint classes and training split D_3 contains noise additional to D_2 samples. Given the three different training sets D_i , which of the samples highlighted in orange within test set D_E qualify as outliers? The underspecification of the problem allows for different conclusions. Example inspired by [43]. 16

- 1.4 Illustration of autoencoder-based manifold learning by the example of monkey face detection: In Fig. 1.4(a), the image is obtained from the natural manifold M and projected into the high dimensional image space. The autoencoder trained to reconstruct monkey face images encodes the image $\mathbf{x}^{(1)}$ by mapping it onto the learned manifold \hat{M} , to subsequently reconstruct $\mathbf{x}^{(1)}$ as $\hat{\mathbf{x}}^{(1)}$ from the low dimensional embedding $\mathbf{h}^{(1)}$. Note that the sampled image lies in \mathbb{R}^n with $n \gg 2$ and the manifold is one-dimensional embedded within \mathbb{R}^2 . In Fig. 1.4(b), the input image does not originate from M and is therefore not placed onto \hat{M} by the encoder e and cannot be reconstructed. 21

2.1 Datastack architecture: *Dataset Factory* commands *Retriever* and *Pre-processor* to download and prepare the dataset, respectively. Note that the preprocessor and iterator deal with arbitrary data formats internally. The interfaces and remaining components work on a byte stream level, thus decoupling the dataset-specific peculiarities from the framework. 27

2.2 Combination of dataset iterators via stacking, splitting, and joining operations: Similar to table views in databases, each higher-level iterator represents a view on its underlying iterator. The stacking, splitting, and joining operations build the foundation for more sophisticated iterators with functionality such as in-memory loading or label mapping. 29

2.3 Use case example on textual outlier detection [22, 99]: Splitting, stacking, joining, label filtering, and in-memory loading is used to represent the full data processing pipeline. The final in-memory iterators are used for training, validation, and evaluation of ML models. 30

2.4 The YAML configuration file specifies the pipeline components, their dependencies, parameterization and the hyperparameter search. Here, the dependency graph of an exemplary deep learning configuration is showcased. To run the training and evaluation pipeline, we instantiate the root components trainer, checkpointing, early stopping and evaluator by passing the dependent components to the constructors via dependency injection. In most cases, a component-specific registry on the lowest dependency level instantiates the respective component. Even though the train and eval components are partially instantiated with the same components, we keep two separate instances to prevent side effects during training/evaluation. 34

2.5 Pipeline component instantiation in blueprint via the component factory: A single experiment config from the YAML config is passed to the blueprint, a data class describing the entire pipeline setup. The blueprint commands the component factory, with its internal (custom) component registry, to instantiate the components defined in the dependency graph within the configuration. We distinguish two types of components here. Firstly, some components can be grouped (e.g., F1 score and accuracy) and do not require individual registration. These components are subsumed within a registry (see Component A Registry which registers Component A.1 and Component A.2). Secondly, standalone pipeline components (e.g., train component) have a dedicated constructable that only instantiates this particular component (see Component B). 34

2.6 Illustration of MLgym system design: The two entry points, *starter* and *warm starter*, forward the blueprints (specification of the experiments) to the multiprocessing gym for execution. As specified in Fig. 2.4, the root and dependency components, highlighted in green and blue, respectively, jointly resemble the dynamically assembled training/evaluation pipeline. The logging environment based on event sourcing and streaming is highlighted in red and supports custom clients, e.g., TensorBoard or Weights&Biases. With the entry points and logging environment, MLgym fulfills rules 5 and 10. 37

2.7 MLboard frontend: a React application that visualizes the training and evaluation progress by subscribing to the websocket API and REST API. 39

3.1	KDD outlier and novelty datasets: Visualization of the train and test splits after reducing the dimensionality to 2D using UMAP [164]. Outlier samples are highlighted in red, inliers in blue and novelties in green. Fig. 3.1(a) and Fig. 3.1(b) show the clustering of the imbalanced variant of datasets, while Fig. 3.1(c) and Fig. 3.1(d) show the clustering for the balanced datasets. Note that, as per definition, novelties only appear in the test set, whereas outliers appear in both.	49
3.2	Imbalanced datasets: Visualization of the Reuters, ATIS, and arrhythmia datasets after projecting the samples onto a two-dimensional plane, using UMAP [164] for dimensionality reduction. Minorities samples are highlighted in red, majorities in blue.	50
3.3	Loss histograms of the best OCA models and ATA models on the Reuters and KDD OI dataset. The explicit reconstruction error maximization of minority samples leads to better separation of majorities and minorities in reconstruction error space. Interestingly, OCA leads to better reconstruction of majorities, but implicitly also minimizes minorities, resulting in poor outlier detection performance.	53
3.4	Conceptual design of a <i>supervised autoencoder</i> (SAE) and a supervised autoencoder with readout loss (SAER). In both variants the autoencoder maps the input sample \mathbf{x} to the reconstruction $\hat{\mathbf{x}}$. In the SAE case, the MLP performs inference solely on the encoding of \mathbf{x} , whereas in the case of SAER, the reconstruction loss is also passed to the MLP, as indicated by the dashed line.	56
3.5	Scatter plots of the minority encodings (red points) and majority encoding (blue points) generated by the best ASAER, ATA, OCA and SAE models on the ARR dataset split by train and test. To visualize the encodings within \mathbb{R}^2 , we employed UMAP, an unsupervised manifold learning based algorithm for nonlinear dimensionality reduction [164]. In contrast to the non-adversarial methods, the results suggest that the adversarial methods successfully separate the minorities from the majority manifold. This becomes especially apparent when comparing the level of separation to the original clustering result on the raw samples in Fig. 3.2(c).	58
3.6	Histograms of reconstruction loss distributions of minority (red bars) and majority samples (blue bars) for the best ASAER, ATA, OCA and SAE models on the ARR dataset split by train and test. Note y-axis is log scaled.	59
4.1	Conceptual illustration of OSR: The closed set $S_{\hat{\varphi}}$ is visualized by the green area comprising all inliers. There are two different rest samples types: red points resemble a known rest class, whereas blue and pink samples reflect unobserved outliers. The objective is to learn an indicator function f (decision boundary shown as blue line), that filters inliers and rejects all rest samples. The open space risk $R_{\mathcal{O}}(f)$ can be interpreted as the ratio of the blue area over the green area that is bounded by the decision boundary of f	64
4.2	Class probabilities of DAE, MLP and MiMo [174] visualized as contours: In contrast to MiMo and MLP, DAE learns a hull around the red class of interest (COI) for the three datasets, enabling the method to not only separate the two inlier classes, but also to reject the unseen uniform noise. MLP and MiMo only wrap the inliers if the rest samples encourage such a decision boundary, as shown for the Bounding Gaussians dataset.	66

4.3 Decoupling Autoencoder (DAE) architecture: A joint architecture composed of an autoencoder $\varphi(\mathbf{x}) = d(e(\mathbf{x}))$ for sample reconstruction, a reconstruction error module e_{MSE} for outlieriness estimation, and an RBF kernel g with standard deviation σ_1 for classification. Thus, the entire network is given by $f(\mathbf{x}) = g(e_{\text{MSE}}(\mathbf{x}, d(e(\mathbf{x}))), \sigma_1)$. . . 69

4.4 Combination of classification objective L_{BCE} and $\|t\|_1$ regularization to optimize for a decision boundary t , which is minimal, but does not compromise classification of inliers (red points) and rest samples (blue points) 70

4.5 Intuition behind the reconstruction error objective L_R : Minimization/maximization of reconstruction losses for inliers (red) and rest samples (blue), respectively. Separation is achieved by weighting the reconstruction losses with w_i for inliers and w_o for rest samples. Therefore, samples on the wrong side of the decision boundary t are heavily optimized. With increasing distance from the decision boundary, optimization decelerates for correctly classified samples, as indicated by the length of the arrows. . . 70

4.6 Reconstruction error landscape of ATA and OCA, visualized on the Half-Moon, XOR Circles, and Bounding Gaussians dataset, equivalent to Fig. 4.2. The color gradient from blue to yellow resembles an increase in the reconstruction error. 82

4.7 Qualitative reconstruction comparison of DAE and autoencoder-based baselines on MNIST₇ and FMNIST_{3,7}: The first row in each of the two grids, shows the original samples. We present three representative inlier samples, followed by three rest samples of the respective splits. In contrast to OCA and C2AE, our DAE method and ATA, with their adversarial loss functions, are able to not only accurately reconstruct inliers, but also obfuscate the rest samples. 84

4.8 Adversarial robustness comparison between DAE and MLP: For each of the splits S_c , S_o , and S_d in the MNIST₇ and FMNIST_{3,7} datasets, the samples (true positives and true negatives with more than 95% confidence) were perturbed via FGSM in 300 steps with a step size of 0.001, resolving the full range from local stability to global adversarial robustness. The results clearly show that DAE is more robust to adversarial perturbations applied to inliers and, especially, rest samples. On the two datasets, DAE is most robust w.r.t. adversarial perturbations on dataset shift samples, demonstrating the advantage of DAE being a true OSR method with bounded open space risk. 85

4.9 Inlier and outlier loss histograms regarding the different cases within the ablation study in Table 4.6. The model output $g(e_{\text{MSE}})$, which maps the reconstruction errors to inlier probabilities is plotted in green, and the decision boundary t is plotted as a dotted red line. Note that in some cases, e.g., in Fig. 4.9(a) to 4.9(d), the maximum reconstruction errors are capped to highlight the accuracy of the decision boundary. The results show that the combination of loss terms \hat{L} leads to inliers and outliers being minimized/maximized, respectively, while the decision boundary being as close to the inliers as possible, maximizes robustness without jeopardizing classification performance. All the other combinations lead to poor classification performance or lack of outlier robustness. Without outlier maximization, similar to the *one-class autoencoder*, the inliers and outliers are not fully separated on S_c and S_o (Fig. 4.9(i) to 4.9(l)). 87

5.1	Uncertainty heatmap of our Informer architecture and the baselines on the half-moon dataset. Informer and DUQ capture aleatoric and epistemic uncertainty whereas the other three ERM baselines only capture aleatoric uncertainty.	91
5.2	Illustration of the Informer architecture for multi-class classification with k classes: An ensemble composed of k Informer components (autoencoder $d_i(e_i(\mathbf{x}))$ for sample reconstruction, a reconstruction error module e_{MSE} for outlieriness estimation, and an RBF kernel g for classification), each learning an OVR relationship for a different inlier class. The uncertainty estimation module (UEM), provides estimates on the total, aleatoric and epistemic uncertainty, as denoted by U_t , U_a and U_e , respectively.	93
5.3	Uncertainty Estimation Module (UEM): Given a prediction \hat{y} , the two class probabilities $p(c_1 \mathbf{x})$ and $p(c_2 \mathbf{x})$, conditioned on the input \mathbf{x} , determine the amount of aleatoric and epistemic uncertainty within the prediction. UEM splits the uncertainty U_t (Manhattan distance of prediction to bottom right corner) into a convex combination of aleatoric uncertainty U_a and epistemic uncertainty U_e . For simplicity, we only show the case $p(\hat{y}_1 \mathbf{x}) > p(\hat{y}_2 \mathbf{x})$	93
5.4	Histogram of IC predictions $\varphi_i(\mathbf{x}) = p(c_i \mathbf{x})$: While the inlier class is generally well-separated from rest classes, there are few outlier samples primarily from split S_o that are falsely predicted as inliers with high confidence.	98
5.5	Comparing the prediction scatter of Informer ($\text{IC}_{\text{pullover}}$ and $\text{IC}_{\text{t-shirt}}$) to DUQ's centroids (C_{pullover} and $C_{\text{t-shirt}}$) w.r.t. inlier classes pullover/t-shirt and S_o/S_{di} rest classes. Due to the majority of samples overlapping in the dense corners, we added histogram bars on the top and right edge for visual support.	99
6.1	Overview of the Extraction Service showing the execution flow of an extraction job through the components in the system. First, a client schedules a document for extraction (1), a job is created and its corresponding PDF is stored (2). A worker then retrieves the job and document (3), parses the document (4), extracts information (5), and builds a JSON representation of the document (6), before sending the document back with its extracted information (7).	108
6.2	Class-specific prediction histograms for each Informer component (IC). Fig. 6.2(a) shows the normalized (i.e., density) reconstruction error histograms for the four ICs, inferred from the test samples. Likewise, Fig. 6.2(b) shows the estimated probability scores, as unnormalized (i.e., absolute frequencies) histograms. Both diagrams show the robustness of the Informer method and explain that the low precision scores in Table 6.4 is caused due to the class imbalance.	113
6.3	The average processing time per page with different thread counts.	114

List of Tables

3.1	Datasets and their subdatasets: Majority and minority frequencies for train, validation and test split. A subclass share of <i>all</i> means that <i>all</i> subclasses are shared between the dataset splits, whereas a subclass share of <i>none</i> , indicates that none are shared between train/val and test split. This is an important requirement for novelty detection. Note that m^+ and m^- refer to the majorities and minorities, respectively.	47
3.2	AUROC and macro F1 scores of the best ATA, MLP, and OCA models on the seven datasets. It is observed that ATA outperforms baseline methods in most of the tasks, depicting its robustness.	53
3.3	Summarization of the different model variations. The models are distinguished on an architectural level based on whether the reconstruction loss is passed to the MLP or not, as defined in the third column. Additionally, there are two different training styles by adaptation of the reconstruction loss function L_R in Eq. (3.14): Mean squared error L_{MSE} and adversarial reconstruction loss L_{R_adv}	56
3.4	Performance comparison of vanilla SAE, ASAE, SAER, and ASAER to the baselines MLP, OCA and ATA based on AUPR scores throughout the seven datasets (ATIS, REU, ARR, KDD OI, KDD OB, KDD NI and KDD NB). For each dataset, the score of the best model has been highlighted in bold face. It is observed that SAE approaches generally outperform the baselines. BASE denotes the expected AUPR scores of a random classifier for reference.	58
4.1	Class assignment within training split S_t and evaluation splits S_c, S_o, S_{d1}, S_{d2} and S_{d3} for the text and image classification datasets, in accordance with [99]: Splits S_c and S_o are representative of tasks T_c and T_o , respectively; splits S_{d1}, S_{d2} , and S_{d3} of T_d . Mapping of rest classes specified in Table 4.2.	78
4.2	Specification of the rest classes for the different dataset splits in Table 4.1, derived from seven different textual and image-based classification datasets.	79

4.3	Performance of DAE and the baselines ATA, OCA, MLP, and MiMo, on splits S_c, S_o, S_{d1}, S_{d2} and S_{d3} of the textual and image datasets with a single COI. The AUROC results are aggregated in the first column for each split, counting best and weak performances. Specifically, when a model’s AUROC score drops at least 5 percentage points below the best AUROC score, the model is counted as weak performing. For reference, BASE denotes the performance of a random classifier that predicts COI with probability $p \sim U[0, 1]$. Metrics and confidence are measured in %. Baseline results on textual datasets adopted from [99]. Across all three subtasks of OSR, DAE and ATA yield the most robust results, while MLP and MiMo show a significant performance degradation on the dataset shift task, similar to OCA on the classification task.	80
4.4	Performance of DAE and the baselines on the derived datasets with more than one COI. Across all tasks DAE and ATA show high robustness, whereas the remaining baselines perform poorly on at least a single task. Baseline results on the Reuters dataset partially adopted from [99].	81
4.5	Calibration of DAE, MLP and MiMo on split S_c in terms of CECE, Brier score and average per-bin calibration error difference ΔCE between T_c and T_o/T_d . All models are similarly well-calibrated on the classification task. The calibration similarity across splits is higher for DAE compared to the other methods. Metrics and confidence are measured in %.	84
4.6	Ablation study on MNIST _{2,7} w.r.t. the loss terms in \hat{L} controlled by hyperparameters λ_i (see Eq. (4.2) and Eq. (4.5)): The respective loss histograms are shown in Fig. 4.9. These results clearly indicate that the combination of all loss terms yields the highest dataset shift robustness, with slight degradation in classification performance. Without the adversarial loss term (i.e., $\lambda_1 = 0$), the models express significant robustness deficiencies and the two cases without L_{BCE} lead to unusable results, as t becomes 0. Note that the F1 scores can deviate from previous results in Table 4.4, as there the best models were selected based on AUROC scores on S_c . If the F1 score is a concern, we suggest filtering models whose decision boundary t has converged to a constant value, and subsequently select the best model based on AUROC.	86
5.1	Three step experiment setup concerning classification (S_c), contextual outlier exposure (S_o) and dataset shift exposure (S_{d1} and S_{d2}): The two sets of inlier/rest classes within a split are denoted the first letter of the original dataset. The training split S_t and test splits (S_c, S_o, S_d) share the same inlier classes for a given dataset. The contextual outlier split S_o and dataset shift splits S_d provide the rest classes from the same dataset and an unrelated dataset, respectively. Note that the sets e and a refer to the classes of EMNIST and ATIS.	96
5.2	Image classification results on the different test splits: The best performing score is highlighted in boldface for each split. Weak performances are highlighted in gray within each split when the score deviates more than 10 percentage points from the best score. The Informer architecture provides competitive classification results while being most robust to outliers/dataset shift.	97

5.3	Results on text datasets with the same score highlighting as in Table 5.2: Similar to the results on image datasets, Informer provides competitive classification without the robustness deficiencies of the other baselines.	97
5.4	Representative samples and their reconstructions from each split, expressing either $\min U_t$, $\max U_e$ or $\max U_a$. The samples clearly show that different uncertainty types can be captured by the Informer architecture.	99
6.1	Number of toxic samples for each of the four test splits. Note, that each test split shares the same 10000 non-toxic samples.	104
6.2	Performance of MLP, MIMO, ATA and OCA on test splits toxic-only, threat, insult and identity-hate. We consider the AUPR and the F1-Score (both with respect to the "toxic" class). As a reference for the base rate dependent metrics, we also report the expected scores for a random classifier (BASE) with uniform probabilities $p \sim U[0, 1]$	105
6.3	The available extractors with their dependencies and machine learning (ML) approaches. The features for each extractor were manually engineered by accounting domain experts.	111
6.4	The performance of document-level classifiers measured in terms of F1 score, precision, and recall. The multi-class classifiers are evaluated in terms of the respective macro scores. Note that the imbalance of the document types within the dataset causes a significant drop in F1 score for the document type extractor, due to the macro averaging.	112
6.5	The performance of entity extractors in terms of accuracy as a "hit-or-miss" metric.	112

Publications

This thesis is based on the following chronologically ordered publications published between 2020 and 2022:

- [1] M. Lübbering, R. Ramamurthy, M. Gebauer, T. Bell, R. Sifa and C. Bauckhage, “From Imbalanced Classification to Supervised Outlier Detection Problems: Adversarially Trained Auto Encoders”, *International Conference on Artificial Neural Networks*, Springer, 2020.
- [2] M. Lübbering, M. Gebauer, R. Ramamurthy, R. Sifa and C. Bauckhage, “Supervised Autoencoder Variants for End to End Anomaly Detection”, *Pattern Recognition. ICPR International Workshops and Challenges*, 2021.
- [3] M. Lübbering, M. Gebauer, R. Ramamurthy, M. Pielka, C. Bauckhage and R. Sifa, “Utilizing Representation Learning for Robust Text Classification Under Datasetshift”, *Proceedings of the Conference "Lernen, Wissen, Daten, Analysen"*, 2021.
- [4] M. Lübbering, M. Pielka, K. Das, M. Gebauer, R. Ramamurthy, C. Bauckhage and R. Sifa, “Toxicity Detection in Online Comments with Limited Data: A Comparative Analysis.”, *ESANN*, 2021.
- [5] M. Lübbering, M. Gebauer, R. Ramamurthy, C. Bauckhage and R. Sifa, “Decoupling Autoencoders for Robust One-vs-Rest Classification”, *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, 2021 1.
- [6] R. Rajkumar, M. Lübbering, T. Bell, M. Gebauer, B. Ulusay, D. Uedelhoven, T. Dilmaghani, R. Loitz, M. Pielka, C. Bauckhage and R. Sifa, “Automatic Indexing of Financial Documents via Information Extraction”, *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2021.
- [7] M. Lübbering, M. Gebauer, R. Ramamurthy, C. Bauckhage and R. Sifa, *Bounding open space risk with decoupling autoencoders in open set recognition*, *International Journal of Data Science and Analytics* (2022) 1.
- [8] M. Lübbering, M. Pielka, I. Henk and R. Sifa, “Datastack: unification of heterogeneous machine learning dataset interfaces”, *2022 IEEE 38th International Conference on Data Engineering Workshops (ICDEW)*, IEEE, 2022 66.
- [9] M. Lübbering, M. Gebauer, R. Ramamurthy, C. Bauckhage and R. Sifa, “From Open Set Recognition Towards Robust Multi-class Classification”, *Artificial Neural Networks and Machine Learning – ICANN 2022*, 2022.