UNIVERSITÄT BONN

Self-Supervised Video Representation Learning and Downstream Applications

DISSERTATION

zur Erlangung des Doktorgrades (*Dr. rer. nat.*)

der Mathematisch-Naturwissenschaftlichen Fakultät

der Rheinischen Friedrich–Wilhelms–Universität, Bonn

vorgelegt von

NADINE BEHRMANN

aus

Bonn, Deutschland

Bonn, 2023

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

1. Gutachter / 1st Advisor: Prof. Dr. Juergen Gall
2. Gutachter / 2nd Advisor: Prof. Dr. Rainer Stiefelhagen
Tag der Promotion / Day of Promotion: 10.03.2023
Erscheinungsjahr / Year of Publication: 2023

Abstract

by Nadine Behrmann

for the degree of

Doctor rerum naturalium

Video understanding is an important computer vision task with a great variety of different applications, such as autonomous driving, robotics and video surveillance. Much progress has recently been made on video understanding tasks related to short trimmed videos, such as recognizing human activities in videos. To a large extent, this progress can be attributed to the existence of large-scale labelled datasets. However, labelling large-scale video datasets quickly becomes prohibitively expensive, especially for tasks involving long-term video understanding of untrimmed videos, such as temporal action segmentation. Here, datasets are still relatively small and methods rely on pretrained video representations. More generally, the way the data is represented has a significant impact on how easy or difficult a task is to solve, and learning better video representations has the potential to greatly facilitate many video understanding tasks. Although supervised learning is the most prevalent representation learning approach, it is prone to miss relevant features, *e.g.* the current popular large-scale datasets are inherently biased towards static features. To alleviate this shortcoming, we explore different self-supervised video representation learning methods. This not only allows us to put an explicit focus on temporal features, but also enables the use of the vast amounts of unlabelled video data available online.

First, we investigate how unobserved past frames can be jointly incorporated with future frames to pose a more challenging pretext task that encourages temporally structured representations. To that end, we propose a bidirectional feature prediction task in a contrastive learning framework that not only requires the model to predict past and future video features but also to distinguish between them via temporal hard negatives. Second, we develop a general contrastive learning method to properly accommodate a set of positives ranked based on the desired similarity to the query clip. As the content of videos changes gradually over time, so should the representations; we achieve this by ranking video clips based on their temporal distance. Third, we develop a method for learning stationary and non-stationary features, motivated by the observation that a diverse set of downstream tasks requires different kinds of features: Stationary features capture global, video-level attributes such as the action class, while non-stationary features are beneficial for tasks that require more fine-grained temporal features such as temporal action segmentation.

Finally, we propose a method for action segmentation that is built on top of pretrained video representations. In contrast to previous works, which are based on framewise predictions, we view action segmentation from a sequence-to-sequence perspective – mapping a sequence of video frames to a sequence of action segments – and design a Transformer-based model that directly predicts the segments.

Keywords: video representation learning, self-supervised learning, action segmentation

Acknowledgements

First, I would like to thank Prof. Juergen Gall for is support and supervision over the past years. I want to thank him for his openness and the freedom to explore research ideas as well as for providing advice and guidance as needed. I also want to thank Prof. Rainer Stiefelhagen who kindly agreed to serve as thesis examiner.

Very special thanks goes to Mehdi Noroozi for his continuous support and supervision, his great ideas and mentorship, his patience and motivation. For the countless hours he spent discussing ideas and supervising me and for being an all-around great supervisor.

Furthermore, I would like to thank my colleagues at the Bosch Center for Artificial Intelligence for a great and productive work environment. It is an honour to be part of this team and I very much enjoyed working with all of you, including Edgar, David, Vadim, Mohsen, Alireza, Dan, Kaspar, Alex, Bill, Nicole and Volker. Thank you for the great collaborations, proof-reading paper drafts and all the fun times and good memories. I'm especially grateful for the amazing collaborations with David, Mohsen and Alireza.

I also want to thank my family and friends for all their support and encouragement. A PhD can be very stressful at times, especially towards deadlines or when combined with a global pandemic, and they were always understanding and supportive and helped me to find a balance. I want to especially thank my parents, who always believed in me and supported me and gave me the opportunity to follow my dreams. Last but not least, I want to thank Jochen for his emotional support and patience.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

An alphabetically sorted list of abbreviations used in the thesis:

BFP	Bidirectional Feature Prediction
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
InfoNCE	Info Noise Contrastive Estimation
LSFD	Long Short View Feature Decomposition
LSTM	Long Short Term Memory Network
OOD	Out-of-Distribution
ResNet	Deep Residual Network
RINCE	Ranking InfoNCE
RNN	Recurrent Neural Network
SCL	Supervised Contrastive Learning
seq2seq	Sequence-to-Sequence
SGD	Stochastic Gradient Lescent
SSL	Self-supervised Learning
UVAST	Unified Video Action Segmentation via Transformers

CHAPTER 1

Introduction

Contents

1.1 Motivation

Videos are becoming more and more prevalent in our day-to-day lives: We use them to share stories and information online, cameras are widely integrated in modern machines, *e.g.* for driver assistance or semi-autonomous driving in our cars, or to observe the surrounding environment of robots, and video surveillance systems increase the security in public and private places and ensure quality control in manufacturing processes. With the steadily growing impact that video applications have on our everyday lives, comprehensive video understanding is becoming an increasingly important computer vision task and has seen increased interest both from industry and academia over the recent years. Video understanding is an umbrella term that encompasses a very broad and general variety of different problems and typically requires video understanding systems to characterize a video through high-level abstract concepts which allow us to answer questions like: What happens where and when in the video? When does an action start and end? Which actors or objects are involved? What has previously happened leading up to this point and what will happen next? How are the current events affecting the surroundings and future events? This wide variety of different tasks requires a variety of different models, each tailored to the specific task at hand. One common underlying question that these systems share is how they can be trained efficiently. A standard approach to reduce both the required amount of labelled data as well as training time is to re-purpose large-scale pretrained models, *e.g.* we can use pretrained weights as an initialization for our current model or we use the pretrained

model as a general-purpose feature extractor and build new models on top of these features. In this setting, we exploit the learned *video representation* of the pretrained model. Generally, the way video content is provided to the models has a large impact on how easy or difficult a task is to solve, and a better, more structured video representation can potentially benefit all of the above tasks.

Modern video representation learning methods are based on deep learning, which – in theory – can solve many complex tasks arbitrarily well. It automatically discovers useful information and relevant features of the input through a number of successive non-linear transformations and has been found to work well for many practical applications as long as abundant data is available. Fortunately, video data is widely available online; in fact, the amount of video content that is uploaded to the Internet every day keeps growing year after year: Between 2014 and 2020 the number of uploaded video content hours on YouTube grew by around 40%, reaching a staggering number of more than 500 hours every minute (*statista*, 2022). Many of the commonly used datasets, *e.g.* Kinetics (*Kay et al.*, 2017), are sourced from online video platforms such as YouTube. In contrast to static images, videos provide an even richer source of information: They show how people and objects interact with each other in the real world and the effect that such interactions have on the environment and future events. These large quantities of video data enable the development of complex video understanding models, leading to tremendous progress in this area. Large-scale models trained on hundreds of thousands of hours of video data already accomplish outstanding breakthroughs on tasks involving short pre-segmented human-centric videos, such as action recognition.

However, these models are trained in a *supervised* fashion, which comes with several major drawbacks: First, labelling large quantities of video data is extremely expensive, posing a major bottleneck for scaling up models and datasets or when tackling new tasks. Second, supervised pre-trained models tend to capture video features that are useful for the task they are trained on and do not necessarily transfer well to tasks that require different features. Both problems can be addressed with *self-supervised* learning. In contrast to supervised learning, which relies on labels to provide a learning signal, self-supervised learning derives a learning signal from the unlabelled data itself. This not only reduces the need for labelled training data drastically, but also allows us to design tasks to target specific features or to encourage the model to explore the structure of the data more thoroughly. For these reasons, self-supervised learning has gained a lot of popularity in many research areas, and achieved major breakthroughs, *e.g.* in natural language processing (*Devlin et al.*, 2019; *Brown et al.*, 2020) and image representation learning (*Chen et al.*, 2020b; *He et al.*, 2020; *Radford et al.*, 2021). In this thesis, we develop several methods to improve self-supervised video representation learning.

While video representation learning addresses the task of extracting high quality features from short clips, videos in the real world usually occur in untrimmed streams of data with several actions happening at various times or (partially) in parallel, potentially separated by irrelevant frames where nothing happens, and related to each other through sophisticated long-range temporal dependencies. Furthermore, with similar background and objects involved in the scene different actions within the same video tend to share many visual cues and are often distinguishable only via subtle variations and motion cues. To make matters worse, the sheer lengths of such untrimmed videos render them impractical for end-to-end training of deep models and instead we need to rely on pre-extracted feature descriptors, *i.e. video representations*, which are trained on short trimmed videos. For these reasons, long-term video understanding remains challenging to this day. Therefore, in this thesis we focus on improving general video representations to get more descriptive video features on the one hand, and developing a method for the long-term video understanding task of action segmentation on

the other.

1.2 Challenges

Increasingly complex video understanding models and tasks intensify the need for large-scale datasets for training. However, the standard approach with fully supervised learning on labelled datasets does not scale well and poses a major bottleneck. Tasks involving long untrimmed videos further amplify this problem. But even when large quantities of labelled data are available, transferability of the learned representations to different tasks and datasets can not be taken for granted: Supervised representations tend to focus more on features that are beneficial for the task they were trained on and are at risk of ignoring other features that are irrelevant for the task. These features can however be crucial for different tasks.

1.2.1 Supervision

The huge quantities of video data that are available on the Internet enable the training of very deep video models. The standard approach to train these models is based on supervised learning and requires labelled data. However, due to the temporal nature of videos the labelling effort is even more expensive than that of images – especially if fine-granular temporal annotations are required – posing a major bottleneck for large-scale labelled datasets. This is further complicated as the provided labels control which types of features the model learns. For example, the commonly used large-scale video datasets consist of a collection of pre-segmented video clips from YouTube, which typically show human actions, and are annotated with coarse-grained action labels. As a result, the model is biased towards static features (*Li et al.*, 2018), *e.g.* the background, which in many cases is sufficient to discriminate between action classes. In fact, previous work has found that in many cases a single to few frames are sufficient for action recognition (*Schindler and van Gool*, 2008; *Wang et al.*, 2019b), with diminishing return when adding more and more frames. Consequently, features learned in a supervised setting can not be expected to transfer well to other tasks that require temporal features.

One obvious way to address this issue is to annotate videos carefully, *i.e.* the labels explicitly reflect temporal information (*Goyal et al.*, 2017) or action classes are annotated in a fine-grained manner (*Shao et al.*, 2020; *Li et al.*, 2018). While such high quality annotations greatly facilitate video understanding, they are much more expensive to obtain and may still bear the risk of neglecting important video features that are not reflected in the labels. Consequently, this would require to re-iterate this process whenever tackling new tasks that rely on different features. However, collecting, refining and labelling new large-scale datasets every time does not scale and would hinder the scientific progress in these fields. A more promising and feasible approach is the development of unsupervised and weakly supervised methods. These methods aim to leverage large quantities of unlabelled data alone or in conjunction with few labelled samples and derive a learning signal that is not based on labels alone. A major advantage of these methods is not only that they circumvent the expensive labelling process, but also provide a convenient tool to learn the distribution of the data more thoroughly and not only through input-output relationships. For these reasons label-efficient methods have become a very popular research area over the recent years and is now one of the subjects with the most published papers at the top tier Computer Vision conferences (*ICCV*, 2021). While

unsupervised learning in its generic form can be applied to almost any domain, in this work we are interested in how to leverage unsupervised learning for general video representation learning.

Research Question 1 *How can we improve video representations without the need for large-scale labelled datasets, which accurately capture the contents of videos and enable a large set of downstream tasks?*

As mentioned above, the short answer to this question seems to be unsupervised video representation learning. In this thesis, we are interested in a specific sub-category, namely, *self-supervised* video representation learning. Self-supervised learning exploits the merits of supervised learning by constructing a pretext task, which does not require any labels and can be trained end-to-end analogously to supervised learning. Current state-of-the-art self-supervised learning methods are based on contrastive learning. While this provides a more scalable approach than supervised learning, contrastive learning was originally designed to learn representation of still images; how to properly adapt it to videos is an open research question to this day.

1.2.2 Temporal Features in Videos

Technically, videos are just sequences of frames, so why can we not simply apply a learned image encoder to each frame independently? In fact, early approaches that apply deep learning on videos were build on images encoders (*Karpathy et al.*, 2014). However, the key difference between images and videos is the temporal dimension: We can observe how actors and objects move and the continuous transformations and deformations that they go through over time. Processing frames individually looses such low-level motion cues. But the problem extends beyond just low-level motion: Many video frames show very visually similar scenes that differ from each other only through small variations that occur over time. Such small variations are especially difficult to extract, but can be crucial for some video understanding tasks. For example, consider fine-grained action classification of gymnastics videos (*Shao et al.*, 2020): two different gymnastics figures flic-flac and salto backward share similar movements – both start with jumping backwards head first, followed by the body turning around in the air and then landing on the feet – and can only be distinguished by subtle differences between them (hands touch the ground in flic-flac but not in salto). Another example can be found in temporal action segmentation: The different steps in preparing a sandwich may look very similar, with the same background and objects in the scene and in order to accurately segment the preparation steps we need to rely on the small variations between "smearing butter" and "smearing jam" or "putting cheese on bread" and "putting cucumber on bread".

As argued above learning fine-grained temporal features in a supervised setting is challenging and opting for unsupervised learning is favorable; however, learning temporal features in a contrastive learning framework can not be taken for granted either. The vanilla contrastive learning methods were initially developed for still images; it is based on pairs of similar and dissimilar images, called *positive* and *negative* pairs, respectively. The positive pair can for example consist of two different augmentations of the same image, while negative pairs are formed with different images. Contrastive learning then trains the model to embed the positive pair close in feature space and simultaneously pushes negative pairs apart. Although this approach can be directly extended to videos, it lacks an explicit focus on temporal features and is still an active research area today. Specifically, it raises the question how we can construct comprehensive positive and negative pairs of video clips to instruct the

model to represent motion cues, small temporal variations, etc. A naive solution simply incorporates temporal augmentations to make positives; however, this biases representations towards temporally invariant features. In fact, it has been found that this can reduce performance in practice (*Bai et al.*, 2020), demanding more advanced solutions, which we will investigate in this thesis.

Research Question 2 *How can we instruct video representation models to capture the temporal structure in videos more accurately? Specifically, how can we better emphasize temporal features in a contrastive learning framework?*

1.2.3 High-Level Temporal Reasoning

Representing temporal variations is especially important for tasks such as temporal action segmentation, which aims to partition a given untrimmed video into the different action instances that occur, assuming each frame belongs to exactly one action. Since background, actor and objects often remain the same throughout the video, the different actions are only distinguishable via small temporal variations. As such, a general purpose video feature extractor that accurately represents such variations can greatly benefit the action segmentation task. This is especially relevant as end-to-end training on the untrimmed videos is too computationally demanding and current action segmentation models rely on pretrained video representations. However, action segmentation goes beyond video representations and requires reasoning at different levels. While the first challenges discussed above evolve around video representations, *i.e.* the "early" levels, other challenges arise when dealing with more complex high-level temporal reasoning that is required for temporal action segmentation. Current state-of-the-art methods treat temporal action segmentation as a framewise classification problem: They predict a single action class for each frame, which define the segmentation over the entire video sequence. One problem with such approaches is that they are sensitive to misclassification errors. While it may seem like having only a few misclassified frames out of thousands is an acceptable rate, they can in fact have a big influence: even few frames can split up an otherwise continuous segment into several chunks, which will then be regarded as several separate actions happening at different times. This phenomenon, which is referred to as *over-segmentation*, can lead to over-counting of actions within a single video and incorrect durations of segments. The current methods try to address this issue from different angles, *e.g.* by employing a smoothing loss on the framewise outputs, or using refinement modules and post-processing algorithms. Ultimately, we are not interested in the classification of individual frames, but rather in the higher-level description telling us which actions happen in the video and when they start and end, raising the question whether framewise predictions are the optimal approach for action segmentation. Although, framewise predictions can be easily transformed to match this format, allowing the model to directly output predictions in the higher-level description space is favorable. By directly predicting the segmentation at a segment-level, such models naturally prevent over-segmentation errors inherent to frame-level predictions. Therefore, our final research question investigates methods that directly map to this higher-level representation of segmentations.

Research Question 3 *Can we develop models for temporal action segmentation which directly map from long untrimmed input sequences to the high-level output space that represents video segmentations at a segment-level rather than frame-level?*

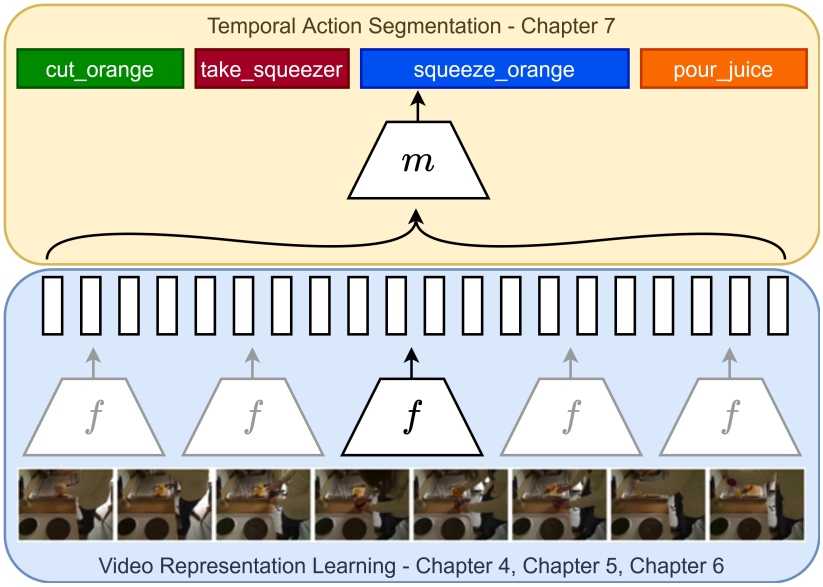


Figure 1.1: **Overview of the Contributions.** Video Understanding requires reasoning at different levels: Our methods in Chapters 4, 5 and 6 aim to extract high quality video representations from the raw input frames, while Chapter 7 introduces a method for temporal action segmentation, a higher-level reasoning task. The former involves learning a video representation model f that extracts video features from short clips; the latter aims to train an action segmentation model m to temporally segment long untrimmed videos and operates on top of the features extracted by f.

1.3 Contributions

Our first challenge in Research Question 1 evolves around the issue with large-scale labelled datasets: Not only is the labelling process of large quantities of video data extremely expensive, but it also does not guarantee that the supervised trained model learns the desired video features. To address this issue, we introduce several self-supervised methods for video representation learning that do not require any labelled data. While there has been tremendous progress in self-supervised image representation learning in recent years – specifically via contrastive learning approaches – self-supervised video representation learning still lags behind, although the additional temporal dimension of video provides even more opportunities for pretext tasks. To address Research Question 1 – how to learn video representations without labelled data – we propose three methods to advance contrastive video representation learning, see Section 1.3.1. Namely, we design methods to better capture temporal features in videos, Research Question 2: First, we exploit unobserved past frames together with future frames for more temporally structured representations, Section 1.3.2. Second, we design a contrastive ranking method to incorporate fine-grained similarities in contrastive learning to better reflect the gradually changing video content, Section 1.3.3. Third, we decompose the representation space into stationary and non-stationary features via contrastive learning, Section 1.3.4. While the first two research questions are concerned with how to learn accurate video representations efficiently, Research Question 3 turns to a higher-level reasoning task: Temporal action segmentation. While current approaches are still based on frame-level classification, we aim to predict the segments directly, Section 1.3.5.

1.3.1 Advancing Contrastive Video Representation Learning

In recent years contrastive learning has advanced the state-of-the-art in self-supervised representation learning in the image domain. The learning signal that is provided to the model, consists of positive and negative pairs – typically a single positive pair, which is *contrasted* with a large amount of negative pairs. The model learns a representation, where positive pairs are embedded close to each other and negative pairs are pushed apart. While positive and negative pairs can be constructed using label information – *e.g.* images of the same class are positives, whereas images of different classes are negatives (*Khosla et al.*, 2020) – contrastive learning is most widely adopted in self-supervised approaches, where pairs are formed without labels: For example applying various augmentations to an image will retain the semantic information; consequently two different augmentations of the same image show the same semantic content in two different *views*. Although the same concept can be applied to video clips, it neglects the temporal dimension – which provides even more opportunities to construct pairs of clips. A straight-forward approach to extend contrastive learning may seem to simply incorporate temporal augmentations – *e.g.* random frame rate or time reversal keep the semantics of a video intact for the most part – into the construction of positive pairs. However, this approach renders the learned representation *invariant* to such temporal augmentation – the model may no longer capture the speed or direction of movements accurately. Instead we propose to exploit the temporal dimension of videos to construct particularly difficult pairs of clips, that force the model to focus on the temporal structure of videos. Our method in Chapter 4 incorporates a set of temporal hard negatives to encourage temporally structures representations, see Section 1.3.2, while our method in Chapter 6 constructs positive pairs to learn stationary and non-stationary features, see Section 1.3.4. The advantages to this approach are two fold: First, since we can design pairs without the need for label information, it allows us to learn video representations in an unsupervised manner and serves Research Question 1 well. Second, it enables us to put explicit focus on temporal features in videos and satisfies Research Question 2. In the following Sections 1.3.2, 1.3.3 and 1.3.4, we discuss the proposed methods in more detail.

1.3.2 Unobserved Past for Temporally Structured Representations

The temporal dimension of videos naturally provide abundant opportunities for constructing challenging pretext tasks. A popular line of research leverages future frames as a supervision signal. Essentially, the idea is that in order to predict what is going to happen in subsequent frames, the model has to reason about a set of present frames in a meaningful way. Future prediction has been explored extensively in the literature with prediction models ranging over various levels: From low-level sensory input where models aim to predict the RGB pixel values of future frames to high-level prediction in feature space, *i.e.* predicting the future in terms of abstract concepts. This unidirectional form of prediction misses out on a second complementary set of frames: the unobserved past. In Chapter 4, we propose to incorporate past frames jointly with future frames to form a more challenging self-supervised task. We argue that the model should not only be able to predict what has happened previously (past prediction) and what is going to happen next (future prediction), but also be able to *distinguish* between past and future. To that end, we predict past and future jointly in a contrastive learning framework and incorporate the reversed order of future and past as a hard temporal negative to encourage temporally structured representations, addressing Research Question 2. Parts of the chapter have been published in (*Behrmann et al.*, 2021b).

1.3.3 Fine-grained Similarities for Contrastive Learning

Traditional contrastive learning assumes a *binary* set of pairs, divided into positives and negatives. In reality, this distinction might not always be possible and current approaches do not allow for samples that are in a gray area. For example, semantically similar classes share visual attributes (*Deselaers and Ferrari*, 2011), such as different dog breeds. Treating them as positives makes the model invariant towards the distinct features of the samples; treating them as negatives hinders the model to exploit their similarities. Another example can be found in contrastive video representation learning: In addition to spatial crops, which are widely used as image augmentations, videos allow to create temporal crops, *i.e.* creating a sample from different frames of the same video. To date, it is an open research question whether temporally different clips of the same video should be treated as positives (*Feichtenhofer et al.*, 2021) or negatives (*Dave et al.*, 2022). Treating them as positives will force the model to be invariant towards the changes over time, but treating them as negatives will encourage the model to ignore the features that remain constant. To address this issue, we propose a method in Chapter 5 that benefits from a fine-grained definition of negatives, positives and various states in between. Key to this is a new contrastive loss, which enforces gradually decreasing similarity with increasing rank of the samples. We apply this method to the video representation learning task and rank temporally closer clips higher than temporally far ones. This encourages the model to reflect the gradually changing nature of video features in the feature space. Parts of this chapter have been published in (*Hoffmann et al.*, 2022).

1.3.4 Stationary and Non-stationary Features

The above methods are designed to learn a single video representation, although there are different types of features in videos – some that remain the same over time and some that vary. We call these features *stationary* and *non-stationary*, respectively. An ideal representation should benefit many different downstream tasks and while some tasks profit more from stationary features, others rely more on non-stationary ones. Stationary features, which remain similar throughout the video, enable the prediction of global, video-level action classes. Non-stationary features on the other hand are more beneficial for downstream tasks that require a more fine-grained temporal understanding, *e.g.* action segmentation. We argue that a single representation to capture both types of features is sub-optimal; in Chapter 6, we propose to decompose the representation space via contrastive learning from long and short views, *i.e.* longer video sequences and their shorter sub-sequences. Stationary features, which remain the same over time, are shared between the long and short views; non-stationary features on the other hand aggregate over time, so the long view should capture an aggregated form of short view features. We construct the positive pairs accordingly: the stationary features of the long view and a single short view form a positive pair used in a stationary contrastive loss; in a non-stationary contrastive loss we use an aggregated form of the short view non-stationary features paired with the non-stationary features of the long view. This enables the model to learn these two complementary types of features in videos without the need for labels and enhances the representation for different downstream tasks and addresses both Research Question 1 and 2. Parts of this chapter have been published in (*Behrmann et al.*, 2021a).

1.3.5 Temporal Action Segmentation via Sequence-to-Sequence Translation

A good video representation forms the basis for many higher-level video understanding tasks, such as temporal action segmentation. As a final contribution, we propose a method for temporal action segmentation in Chapter 7, which can be applied in the fully supervised and timestamp supervised setting. In contrast to current state-of-the-art frame-level prediction methods, we view action segmentation from a sequence-to-sequence translation perspective, *i.e.* mapping a sequence of video frames to a sequence of higher-level action segments, addressing Research Question 3. Our proposed method involves a series of modifications and auxiliary loss functions on the standard Transformer (*Vaswani et al.*, 2017) model to cope with the strong mismatch between long input sequences opposed to short output sequences and relatively few videos to learn from. We incorporate an auxiliary supervision signal for the encoder via a frame-wise loss and propose a separate alignment decoder for an implicit duration prediction. Parts of this chapter have been published in (*Behrmann et al.*, 2022).

1.4 Publications

Parts of this thesis are based on the following publications:

- **Unsupervised Video Representation Learning by Bidirectional Feature Prediction**
  Nadine Behrmann, Juergen Gall, and Mehdi Noroozi
  IEEE Winter Conference on Applications of Computer Vision (WACV), 2021.

- **Long Short View Feature Decomposition via Contrastive Video Representation Learning**
  Nadine Behrmann, Mohsen Fayyaz, Juergen Gall, and Mehdi Noroozi
  IEEE International Conference on Computer Vision (ICCV), 2021.

- **Ranking Info Noise Contrastive Estimation: Boosting Contrastive Learning via Ranked Positives**
  David Hoffmann*, Nadine Behrmann*, Juergen Gall, Thomas Brox, and Mehdi Noroozi
  The AAAI Conference on Artificial Intelligence (AAAI), 2022.
  doi: 10.1609/aaai.v36i1.19972.

- **Unified Fully and Timestamp Supervised Temporal Action Segmentation via Sequence to Sequence Translation**
  Nadine Behrmann*, S. Alireza Golestaneh*, Zico Kolter, Juergen Gall, and Mehdi Noroozi
  European Conference on Computer Vision (ECCV), 2022.
  doi: 10.1007/978-3-031-19833-5_4.

(* denotes equal contribution)

1.5 Structure

In this thesis, we develop new methods for self-supervised video representation learning, specifically using contrastive learning; we investigate the role of hard temporal negatives as well as hard pos-

itives for contrastive video representation learning and develop a method for samples in-between. While these methods improve the overall representation of video models, our last contribution involves a more high-level temporal modelling task, namely, we develop a method for temporal action segmentation. The thesis is organized as follows.

In **Chapter 2**, we discuss literature related to this thesis, starting from self-supervised video representation learning – and contrastive learning in particular – moving to recent works on temporal action segmentation.

In **Chapter 3**, we define the video representation learning task formally and introduce the necessary tools used throughout the thesis.

In **Chapter 4**, we propose a method for contrastive video representation learning based on bidirectional feature prediction with temporal hard negatives. While *future prediction* constitutes an established and popular direction for self-supervision, the *unobserved past* has been largely overlooked. We argue that past clips provide an equally valuable source of supervision and demonstrate how both future and past can be jointly incorporated into a comprehensively challenging task – which not only requires the model to predict past and future, but also to distinguish between them via a temporal hard negative constructed from the inverse order of future and past.

Positive and negative pairs play a vital role in contrastive learning; however, due to its binary nature it leaves samples that lie in-between – samples, which can be neither identified as positive nor negative – unaddressed. In **Chapter 5**, we develop a method for such cases, allowing a ranked set of positives, which is mirrored in the representation space. Our method not only allows for, but also benefits from such gray-area-samples; we demonstrate the effectiveness of our approach on video representation learning, where positives are ranked based on their temporal distance.

In **Chapter 6**, we identify two different types of features in videos: stationary and non-stationary features, which are beneficial for different types of downstream tasks. We propose to decompose the representation space into stationary and non-stationary features via contrastive learning from long and short views, *i.e.* long video sequences and their shorter sub-sequences. Stationary features are shared between the short and long views, while non-stationary features aggregate the short views to match the corresponding long view; we construct the positive pairs accordingly.

In **Chapter 7**, we propose a model for temporal action segmentation. In contrast to the current state-of-the-art *frame-wise* prediction models, our method aims to predict the segments directly. Taking a sequence-to-sequence perspective – mapping a sequence of video frames to a sequence of action segments – we propose a Transformer (*Vaswani et al.*, 2017) based model. The strong mismatch between input sequence lengths and output sequence lengths – many highly similar input frames opposed to only few output segments – makes this task challenging for a vanilla Transformer model and calls for several modifications and auxiliary losses: To provide more direct feedback to the encoder we apply a frame-wise classification loss on top of the encoder features, and since predicting the durations of segments explicitly remains challenging, we propose an implicit form of duration prediction via our proposed alignment decoder.

Finally, **Chapter 8** concludes the thesis with a discussion of the addressed challenges in contrastive video representation learning and temporal action segmentation, and an outlook on future research.

CHAPTER 2

Related Work

In this chapter, we discuss the most related literature to this thesis. Comprehensive video understanding requires reasoning at different levels: On the one hand, it requires understanding of the raw input frames. This typically involves extracting high quality features, *i.e.* video representations, of short clips within the video that accurately capture the content of the frames, such as low-level appearance and motion cues, objects and scenes. On the other hand, it requires long-term temporal reasoning as videos can potentially span over long time horizons. In this thesis, we consider the task of temporal action segmentation of long untrimmed videos. In the following, we will review previous approaches in both categories, *i.e.* video representation learning and temporal action segmentation. As the commonly used architectures for video representation learning were initially designed for action recognition, we will also give a brief overview of action recognition models as well as the datasets used throughout the thesis.

Contents

2.1 Video Representation Learning

Video representation learning lies at the core of comprehensive video understanding. An ideal representation extracts useful information that benefits numerous downstream tasks such as action recognition, video retrieval, temporal action segmentation or detection and many more. While early approaches represent videos via hand-crafted features, *e.g.* improved dense trajectory (IDT) features (*Wang and Schmid*, 2013), modern methods extract features using a *learned* video model, *e.g.* a convolutional neural network (CNN) such as I3D (*Carreira and Zisserman*, 2017), 3D-ResNet (*Hara et al.*, 2018), or SlowFast (*Feichtenhofer et al.*, 2019). Most commonly, such models are pretrained

in a supervised setting, which typically requires large-scale labelled datasets, such as Kinetics (*Kay et al.*, 2017). However, there are several major drawbacks with this approach: First, it requires huge amounts of labelled data, which are expensive to obtain. And second, even when large quantities of labelled data are available, an ideal representation can not be taken for granted: the learned representations will tend to capture the features, which are useful for the dataset the model was trained on – and many of the datasets available are inherently biased towards static features (*Li et al.*, 2018). This is especially problematic for more complex downstream tasks (such as temporal action segmentation), which require temporal attributes of videos. One way to address this issue is to provide more informative labels, *i.e.* very fine-grained labels that force the model to explore the temporal dynamics of videos, *e.g.* FineGym (*Shao et al.*, 2020), or explicitly labelled temporal movements such as the Something-Something dataset (*Goyal et al.*, 2017). Another way is through *self-supervised* representation learning, which does not require any labelled data and instead derives a supervision signal from the data itself. This approach has a major advantage: Rather than collecting and labelling huge and extremely expensive datasets to learn new types of features that were omitted with previous datasets, we can construct a self-supervised task that forces the model to explore that structure of the data more thoroughly and extract the desired features. For these reasons, self-supervised learning has recently seen increased interest by the community and tremendous progress has been achieved. We broadly categorize current methods into two groups: pretext task based methods and contrastive methods. We will review a selection of methods from these two categories. For a more comprehensive overview we refer to (*Jing and Tian*, 2020).

2.1.1 Pretext Tasks for Self-Supervised Video Representation Learning

Extending Image-Level Pretext Tasks to Videos. One straightforward option is to repurpose image-level pretext tasks, which have proven to be very effective for representation learning, and extend them to videos. For example, *Jing and Tian* (2018) apply the rotation task proposed in (*Gidaris et al.*, 2018) to videos; namely, they rotate video clips by 0°, 90°, 180°, or 270° and then ask the model to predict which rotation was applied. *Kim et al.* (2019), on the other hand, construct a space-time cubic puzzle, analogous to the jigsaw puzzle task (*Noroozi and Favaro*, 2016) for images, and train a model to predict which permutation was applied to the sequence of 3D spatio-temporal crops. While these works demonstrate that image-level pretext tasks can be successfully used to learn video representations, the temporal structure in videos provides further opportunities for a more comprehensive set of pretext tasks.

Temporal Pretext Tasks. Early works derive a supervision signal from the *order of frames*. *Misra et al.* (2016) pose a temporal order verification task: They shuffle a set of video frames and then train a model to verify whether the frames are ordered correctly or not. Similarly, *Lee et al.* (2017) shuffle a set of frames and predict their order. On the other hand, *Xu et al.* (2019) propose to shuffle a set of video clips rather than frames to maintain the video dynamics of the input to a 3D CNN. *Fernando et al.* (2017) propose Odd-One-Out Networks: They ask the model to predict the odd video sequence, *i.e.* a video sequence with wrong temporal order of video frames, from a set of otherwise intact video sequences (which maintain the correct temporal order). *Wei et al.* (2018) construct a pretext task from the arrow of time, *i.e.* they ask the model to recognize whether a video is playing forwards or backwards in time. This is a particularly challenging task as low-level motion patterns

are often reversible in time, *e.g.* moving left to right vs. moving right to left, and requires higher-level reasoning, such as water flows downward and people typically walk forwards rather then backwards, to determine the direction of time.

Other methods exploit *ego-motion* (*Jayaraman and Grauman*, 2015; *Agrawal et al.*, 2015): Namely, *Jayaraman and Grauman* (2015) enforce equivariance of the learned features tied to ego-motion, *i.e.* the distinct transformations that are induced by ego-motion should be mirrored in the feature space, whereas *Agrawal et al.* (2015) learn to predict the ego-motion information given the input frames.

Temporal coherence in videos provides further opportunities to construct pretext tasks; essentially, (spatio-)temporal coherence describes the fact that video content varies smoothly over time (and space). *Wiskott and Sejnowski* (2002) propose slow feature analysis to extract slowly varying features by minimizing the temporal gradient of the representation. *Wang and Gupta* (2015) observe that two patches of frames that are connected via a track should have similar visual representations and use an unsupervised tracking algorithm to obtain such patches. They train a Siamese triplet network using a ranking loss, where the distance between the original and tracked patch should be smaller than the distance to a random patch. *Isola et al.* (2016) learn a model to predict the spatial or temporal proximity in images and video frames, respectively. They then cluster the predicted co-occurrences to obtain groupings of visual primitives and discover that these groupings inherit semantics such as a group of patches corresponds to an object and a group of frames comprises a movie scene. While most methods consider temporal coherence in the first-order derivative, *i.e.* temporally close frames should exhibit only small changes changing slowly over time, *Jayaraman and Grauman* (2016) generalize this notion to higher-order derivatives to encourage steadiness, *i.e.* feature changes are smooth over time. *Lai and Xie* (2019) propose a reconstruction based task, where the model is asked to reconstruct a target frame by pointing to the corresponding pixels from a reference frame. The learned pixel-wise correspondence can for example be applied to video object segmentation and keypoint tracking. *Vondrick et al.* (2018) show that the temporal coherency of color enables the learning of a tracking algorithm, whereas *Wang et al.* (2019c) use the cycle-consistency of time to learn correspondence. *Luo et al.* (2020) create a video cloze procedure inspired by the reading comprehension test, where participants are asked to fill in blank spaces in a text given a list of words to choose from. Similarly, *Luo et al.* (2020) blank out video clips and then create options from the held out clips by applying spatio-temporal transformations. Given the surrounding temporal context, the model then has to pick out the correct option from those clips.

Another line of work is based on the *speed* of videos: Essentially, the underlying assumption is that in order to detect or predict the synthetically altered frame rate the model needs to learn high quality motion features and semantics. For example, speeding up a video of a person walking will not look like the person is running, although they are moving across the frame at the same pace; analogously, slowing down a running video will not turn into a walking video. Thus, the model needs to learn distinct motion features to discriminate between a sped up walking video and a slowed down running video. Furthermore, as argued in (*Epstein et al.*, 2020), predicting video speed requires the model to take at least several frames of the video into account as a single frame is not sufficient to predict speed, and learn features that correlate with the expected duration of actions. *Benaim et al.* (2020) train a binary classifier to distinguish between videos at normal speed and videos that are sped up ($2\times$). They show how the learned model can be used to adaptively speed up test videos: as long as the model does not recognize the video segment as "sped up" they keep speeding it up

further. *Epstein et al.* (2020) on the other hand ask the model to discriminate between four different frame rates and demonstrate the effectiveness of their approach for recognizing unintentional actions. *Wang et al.* (2020b) combine the pace prediction task with a contrastive loss, where two clips of the same video with different pace are used as positive pair, while *Yao et al.* (2020) combine the speed prediction task with a reconstruction decoder that reconstructs the original video clip given the feature of the temporally downsampled version. *Jenni et al.* (2020) incorporate speed prediction as one component in a more general temporal transformation prediction task, which additionally includes random permutation of frames, a periodic arrangement of forwards and backwards direction, and a warping transformation that leaves a random number of frames between every pair of selected ones.

Future Prediction. Another important category of self-supervised learning methods involves dividing the data into two partitions, and training a convolutional neural network (CNN) that *predicts* one part given the other. The key ingredients for these approaches are the design of a partitioning paradigm and the definition of a loss function to quantify the prediction. For instance, the task of image colorization (*Zhang et al.*, 2016; *Larsson et al.*, 2016) proposes to divide the input image into Lab channels and train a model that predicts ab channels given L channel. In the video domain, partitioning in temporal direction has been intensively explored (*Lotter et al.*, 2017; *Mathieu et al.*, 2016; *Srivastava et al.*, 2015; *Vondrick et al.*, 2016b).

One specifically popular partitioning paradigm – inspired by the *predictive coding* theory in neuroscience – takes the form of *future prediction*. Here, the underlying idea is that in order to predict what will happen in the future, the model needs to develop an internal representation of the current state of the world, how entities behave over time and what types of transformations object can undergo (*Lotter et al.*, 2017). A straight forward approach to quantify prediction is based on raw data, where the target distribution is fixed as true data distribution. A naive approach includes reconstruction losses, which are based on strong assumptions and consequently do not achieve decent results. For instance, ℓ_2 assumes the data follows a Gaussian distribution while cross-entropy provides a discrete approximation of the data distribution. To eliminate these limitations, an adversarial loss has been employed in (*Pathak et al.*, 2016; *Mathieu et al.*, 2016). However, the supervision signals arising from true data distributions suffer from ambiguity, *e.g.* in the form of noise or uncertainty, and require the network to devote substantial capacity to model a data distribution, which is not necessarily the optimal solution for representation learning. Addressing this problem, *Vondrick et al.* (2016a) suggest to predict the representation of future frames instead of pixel values, and use a mixture model to handle the multi-modal distribution of future representations. Recently, predicting features of the future video blocks in a *contrastive* fashion has gotten great attention (*Han et al.*, 2019, 2020b), achieving remarkably better performance. The idea is to perform prediction in a learnable transformation of raw data trained jointly with the prediction quantification. More specifically, *Han et al.* (2019) train a 3D CNN that takes a sequence of video clips and predicts the features of a future clip. They apply the same backbone to extract features from present and future clips and quantify prediction via the *InfoNCE* loss (*van den Oord et al.*, 2018), which involves maximizing an estimate of mutual information (*Poole et al.*, 2019) and has been successfully applied in multiple domains (*Chen et al.*, 2020b). In an extension of their work, *Han et al.* (2020b) very recently propose a memory-augmented version and consider both optical flow and RGB videos as input modalities. *Surís et al.* (2021) learn the predictability of the future; namely, their loss is based on the one in (*Han et al.*, 2019) but applied in hyperbolic space – which is well suited to model hierarchical concepts

– allowing the model to manage the uncertainty inherent to future prediction in a more natural way. When the model is certain it will predict the future at a more concrete level, when it is less certain it defaults to a higher-level of abstraction. In contrast to these methods based on *unidirectional* feature prediction, we propose *bidirectionally* predicting future and past features in Chapter 4.

2.1.2 Contrastive Representation Learning

Contrastive learning has recently advanced the state-of-the-art in self-supervised representation learning, closing the gap between unsupervised and supervised pretraining on many image understanding tasks. The proposed methods in Chapters 4, 5 and 6 are based on a contrastive loss, addressing different aspects of contrastive learning for video representations. Here, we provide an overview of contrastive learning methods in general – many of which were designed for the image domain – before inspecting contrastive video representation learning in more detail in Section 2.1.3.

Mutual Information for Representation Learning. The commonly used *InfoNCE* loss (*van den Oord et al.*, 2018) gained popularity in 2018. *van den Oord et al.* (2018) initially used it for contrastive predictive coding to sequentially predict the representations of image patches or audio signals, aiming to learn a feature space that captures information of the input data that is maximally informative to predict future samples. To that end they propose to maximize the mutual information between the present and future samples via the InfoNCE loss, which they show to be a lower bound on the mutual information. Indeed, the idea to maximize mutual information in order to learn useful representations has been explored in other works as well: *Hjelm et al.* (2019) maximize the mutual information between input data and the learned representations at a global (complete input) or local scale, *i.e.* the average mutual information between the representation and local regions of the input. They investigate different estimators of mutual information including the MINE estimator (*Belghazi et al.*, 2018), Jensen-Shannon-Divergence and InfoNCE. *Bachman et al.* (2019) extend the work of (*Hjelm et al.*, 2019) using a more powerful encoder architecture to maximize the mutual information between multiple feature scales instead of a single global and local one and applying it to two independently augmented versions of an image rather than the same single image.

However, the rationale behind these approaches, namely that mutual information yields useful representations, has been challenged by *Tschannen et al.* (2020b): While *van den Oord et al.* (2018) propose that the InfoNCE loss optimizes a lower bound on the mutual information, *Tschannen et al.* (2020b) find that it violates the assumptions in practice and further show that optimizing a more accurate approximation of mutual information in fact hurts the representation quality. Nevertheless, InfoNCE works well in practice, suggesting that its good empirical performance goes beyond mutual information maximization alone. In fact, *Tschannen et al.* (2020b) further establish the connection of the InfoNCE loss to deep metric learning, and specifically the *N-pair* loss (*Sohn*, 2016), and advocate that an interpretation from this angle gives a more plausible explanation. Indeed, more recent work moves away from mutual information perspective and repurpose the InfoNCE loss for an instance recognition task.

Instance Recognition. The idea of instance recognition has become the underlying principle for many modern contrastive learning methods. Instance recognition is rooted in the intuitive idea that the semantics of an image remains intact through typical variations in the data, which are simulated

by randomly applying transformations (*Dosovitskiy et al.*, 2016). *Dosovitskiy et al.* (2016) construct surrogate classes, one for each image, and treat instance recognition as a classification task.

However, this approach quickly becomes infeasible with larger datasets as the number of surrogate classes scales with the number of samples in the dataset. Therefore, more recent work relies on a contrastive loss to distinguish between samples generated from the same instance and samples of different instances. In this scenario the model does not need to identify *which* sample is presented but only if two samples originate from the same instance. More specifically, it involves similar and dissimilar pairs, commonly referred to as positive and negative pairs, and aims to maximize the similarity of positive pairs in feature space while simultaneously minimizing the similarity of negative pairs using the InfoNCE loss. These pairs are typically obtained through different *views* of the same data, *e.g.* by applying two different augmentations (*Chen et al.*, 2020b) to the same image to observe an image under different "viewing" conditions. In fact, this augmentation-based approach has lead to major breakthroughs in self-supervised image representation learning. *Chen et al.* (2020b) further improve the InfoNCE loss in several ways. First, they show that applying a learnable nonlinear transformation before the contrastive loss is superior to a linear transformation. Furthermore, they demonstrate that normalizing the representations and using a temperature parameter to compute the similarity scores is beneficial.

Other common approaches to obtain views for contrastive learning includes different channels or multi-modal data (*Tian et al.*, 2020a), counting (*Noroozi et al.*, 2017), permutation (*Misra and van der Maaten*, 2020), or augmentations (*Chen et al.*, 2020b). The augmentations used to construct views, such as those explored in (*Chen et al.*, 2020b), have a substantial impact on the learned representation and determine which transformations the learned representation becomes invariant to. For example, heavy cropping leads to occlusion-invariant representations (*Purushwalkam and Gupta*, 2020). Selecting the set of augmentations to form views is therefore one critical aspect in contrastive learning; ideally, the augmentations should make the task to identify the correct image pairs difficult enough to prevent shortcuts, while maintaining semantics and important image features. *Chen et al.* (2020b) determine the optimal set of (hand engineered) augmentations via extensive evaluations on downstream tasks, which require labelled data and may not be applicable in setting where labelled data is scarce. Therefore, *Reed et al.* (2021) propose to select the set of augmentations based on the downstream performance on an unsupervised proxy task that is highly correlated with the target task (*e.g.* predicting image rotations) and develop a practical algorithm for automatic selection of augmentations, including the strength of the augmentations and the probability of applying them. Nevertheless, the optimal invariances induced by the augmentations used for contrastive learning may be task-specific. *Xiao et al.* (2021b) propose an alternative framework in which the model learns to capture both varying and invariant factors. To that end, they map the representations to several separate embedding spaces, where embeddings are invariant to all but one augmentation, *i.e.* an augmented sample serves as a positive in one embedding space (invariant to the augmentation) and as a negative in another (variant to the augmentation).

Negative pairs. The negative pairs, which are traditionally obtained by taking different samples in the batch, play a vital role in contrastive learning as they prevent shortcuts via low-level image statistics (*e.g.* edges, corners, etc.) and collapsed solutions. Therefore, having access to a large number of negatives increases the chances to observe samples that are more difficult to distinguish. *Chen et al.* (2020b) verify this intuition by demonstrating the gain (very) large batch sizes bring.

However, such large batch sizes (*e.g.* 4096) are very computationally demanding. Therefore, *He et al.* (2020) propose an alternative approach: They propose to keep features of past batches in a memory bank, which enables the storage of a large set of negatives. To ensure that features in the memory bank are changing smoothly during the optimization process, they propose to use a momentum encoder of the original network to extract features. Other approaches obtain difficult negative samples through hard negative mining (*Robinson et al.*, 2021) or by constructing them explicitly: For example, *Han et al.* (2019) and the follow-up work (*Han et al.*, 2020b) obtain hard negatives from different spatio-temporal locations in the feature map, while *van den Oord et al.* (2018) use different patches in the same image as negatives. Our method in Chapter 4 investigates the role of hard temporal negatives for contrastive video representation learning.

Positive Pairs. Positive pairs provide our model the information which samples should be represented similarly in feature space. In order to provide an effective learning signal, the positive pair should be sufficiently difficult to identify, otherwise, the model might learn a degenerate solution that relies on shortcuts such as low-level statistics. While the widely adopted instance recognition achieves this via strong augmentations independently applied to the same instance, other methods use positive samples that are naturally difficult to identify, *e.g.* pairing videos with audio (*Patrick et al.*, 2021) or images with text (*Radford et al.*, 2021), or designing hard positives in some form. For example, *Zhu et al.* (2021) synthesize hard positives in feature space by extrapolating from true positive samples. Our method in Chapter 6 can be interpreted similarly: By pairing long views with short or aggregated views, the difficulty of the task lies within the positive pairs, which require the model to make a connection in terms of stationary and non-stationary features.

In instance recognition, the positive pairs are obtained from the same instance, while different instances serve as negatives even when they share the same semantics. As a result, the learned representation will aim to be invariant to instance-level variations. On the other hand, *Wang et al.* (2020a) propose to learn representations invariant to category-level variations, *i.e.* pairing samples with different instances of the same category. As the category information is not available in an unsupervised setting, they propose *invariance propagation*: Here, positive samples are discovered by recursively adding the k nearest neighbors of the positives in feature space, *i.e.* the samples that are connected to the true positive through high-density regions. Under the assumption that the label function is smooth in high density regions, the samples discovered through this process have the same label but show higher intra-class variations. In practice, they select positives with the lowest similarity to the query to obtain hard positives. Other approaches obtain additional positives by sampling the nearest neighbors in feature space (*Dwibedi et al.*, 2021) or by clustering the feature space: *Li et al.* (2021a) use *prototypes* in the clustered feature space – representative embeddings for a cluster of similar instances – and use them as positives to enforce each sample to be similar to its corresponding prototype. *Caron et al.* (2020) use the prototypes of one view as the positive for the other view. *Han et al.* (2020a) and *Tokmakov et al.* (2020) augment the same-instance positives with instances belonging to the same cluster in a different feature space. *Khosla et al.* (2020) use class labels to define a set of positives. False negatives are eliminated from the InfoNCE loss by *Huynh et al.* (2022), either using labels or a heuristic. Integrating multiple positives in contrastive learning is not straightforward: the set of positives can be noisy and include some samples that are more related than others. In Chapter 5, we provide a method to properly incorporate such samples.

Supervised Contrastive Learning. Although contrastive learning is mainly used in an unsupervised setting, some recent works incorporate labelled training data into the framework. *Romijnders et al.* (2021) use pseudo labels obtained from a detector, *Tian et al.* (2020c) use labels to construct better views and *Neill and Bollegala* (2021) use the similarity of class word embeddings to draw hard negatives. The term *Supervised Contrastive Learning* (SCL) is introduced in (*Khosla et al.*, 2020) showing that SCL outperforms standard cross-entropy. In the SCL setting ground truth labels are available and can be used to define positives and negatives. Commonly, samples from the same class are treated as positive, while instances from all other classes are treated as negatives. *Khosla et al.* (2020) find that the SCL loss function outperforms cross-entropy in the supervised setting. In contrast, *Huynh et al.* (2022) aim for an unsupervised detection of false negatives. They propose to only eliminate false negatives from the InfoNCE loss which leads to best results for noisy labels.

Use Cases of Contrastive Learning. In principle, representation learning aims to extract features that are useful for a diverse set of downstream tasks. *Ericsson et al.* (2021) evaluate a selection of popular self-supervised image representation learning methods on 40 different downstream tasks covering a wide range, from classification tasks (including texture, scene and fine- and coarse-grained object classification) to few-shot recognition, object detection and dense prediction tasks. Overall, they find that self-supervised pretrained models outperform supervised pretraining on most tasks, especially on few-shot, object detection and dense prediction tasks. Nevertheless, there is no single best model that outperforms all others, suggesting that a universal pretrained model has not yet been realized. Other works extend or reuse the contrastive learning framework for different applications or problems: *Chen et al.* (2020c) find that models pretrained with the contrastive method of (*Chen et al.*, 2020b) are strong semi-supervised learners – especially when very big models are used. More specifically, they use the task-agnostic self-supervised pretrained model, finetune it on a very small amount of labelled data and then distill the predictions to a student network. *Azizi et al.* (2021) show the effectiveness of contrastive pretraining for medical image analysis, specifically for classification of dermatology conditions from images and pathologies from chest X-rays, while *Chaitanya et al.* (2020) extend the contrastive learning framework for segmentation of MRI images in a semi-supervised setting. *Tian et al.* (2020b) use the contrastive loss to distill representations of one model to another. *Winkens et al.* (2020) find that InfoNCE loss is better suited for out-of-distribution detection than cross-entropy. *Chen et al.* (2021b) learn shot embeddings and transfer them to the task of scene boundary detection in movies. A shot – an uninterrupted series of frames captured by the same camera – is relatively easy to localize using low-level cues, while scenes can consist of visually distinct shots and are much more challenging to localize. *Chen et al.* (2021b) use similar shots within a neighborhood as positives to learn shot embeddings in a fully unsupervised contrastive setting. Afterwards, an MLP is trained on top of the frozen shot embeddings in a supervised setting. *Zhang et al.* (2022) propose a temporal equivariant contrastive learning paradigm to learn representations for temporal action localization in videos.

2.1.3 Contrastive Learning for Video Representations

The main difference between images and videos is the temporal dimension, and while extending contrastive learning to the video domain is just as easily as image pretext tasks, the question of how we can best make use of temporal information remains valid here as well.

Contrastive Learning with Temporal Pretext Tasks. A naive attempt includes temporal augmentations into the set of augmentations used to construct positive pairs; however, it turns out that in practice this approach deteriorates the performance on downstream tasks (*Bai et al.*, 2020). Some approaches have combined the contrastive loss with temporal pretext tasks. For example, *Bai et al.* (2020) combine temporal augmentation in contrastive learning with predicting the applied transformation using a separate prediction head. Similarly, *Wang et al.* (2020b) combine the pace prediction task with a contrastive loss in which positive pairs are consist of clips from the same video with different pace. *Jenni and Jin* (2021) combine temporal pretext tasks with two contrastive objectives for instance recognition and time equivariance. *Yao et al.* (2021) combine an intra-frame and and inter-frame contrastive loss with a temporal order verification task. Some of these works also experiment with transformation-based positives, *i.e.* pairing clips of two *different* videos as positives when the same temporal transformation was applied. For example, *Wang et al.* (2020b) find that "same pace" positives degrades representation quality. *Jenni and Jin* (2021) explore a different approach based on equivariance: They encode the *relative* temporal transformation between two clips and pair them with the same relative transformation feature of a different video. In both cases, the "content agnostic" contrastive loss does not work on its own and has to be paired with other self-supervised losses. In summary, these approaches find that adding temporal pretext tasks to contrastive learning improves the quality of the representations, which demonstrates that "vanilla" contrastive learning – *i.e.* only relying on image augmentations and disregarding the temporal dimension – lacks a comprehensive understanding of the temporal information in videos.

Temporal Augmentations. One determining factor for the characteristics of the learned representations originates in the set of augmentations; thus, incorporating some form of temporal augmentations for video representation learning may seem like the natural step forward. *Qian et al.* (2021b) investigate the importance of spatial and temporal augmentations for contrastive video representation learning. While they find that both spatial and temporal augmentations are effective, maintaining temporal consistency within the spatial augmentations (*e.g.* for a random crop the same patch location should be taken from all frames) is crucial. Their temporal augmentations involve sampling the time interval between the positive pairs of video clips and they find that decreasing the probability of larger intervals is favorable over uniform sampling. The frame sampling mechanism in (*Pan et al.*, 2021) can be interpreted as a learned form of temporal transformations: They learn a model to adaptively drop out the "most imporant" frames of a video sequence using adversarial learning. *Kuang et al.* (2021) extend the framework of (*Yao et al.*, 2021) with a global, video-level contrastive loss where the positive pair consists of two sets of frames sparsely sampled over the entire video. *Feichtenhofer et al.* (2021) sample positive clips that are shifted in time and evaluate their approach in several self-supervised learning settings including MoCo (*He et al.*, 2020), SimCLR (*Chen et al.*, 2020b), BYOL (*Grill et al.*, 2020) and SwAV (*Caron et al.*, 2020). *Yang et al.* (2020) sample positive pairs of fast and slow clips and enforce similarity between the representations at several different layers in the network. *Wang et al.* (2022) propose to use long and short views as positive pairs by using different temporal strides, which effectively results in clips of different speeds. On the other hand, *Recasens et al.* (2021) use long and short views but reduce the spatial resolution of the long view instead of the temporal resolution; they only evaluate their method in a non-contrastive setting. Although the above approaches work well on the considered downstream tasks (*e.g.* action recognition), using temporal augmentations such as temporal shift aims to make the model invariant to

these augmentations. This may bias the representations towards temporally persistent features that remain the same over time and potentially loose more fine-grained variations that occur between the different clips. To overcome this issue, *Sun et al.* (2021) propose an *augmentation-aware* contrastive learning framework: many data augmentations are parameterizable, *e.g.* shifts in space and time can be described by the difference between pixel coordinates or time indices, and can be encoded to provide information about the applied augmentations to the projection head. In this way, the contrastive framework can take the transformation information into account when aligning the different views in feature space. *Qian et al.* (2022) identify features of local clips with the corresponding location in the feature map of the global video to form positive pairs.

Hard Negatives. Simply maximizing the similarity of two differently augmented pairs of features does not pose a challenging task – the model can simply assign all samples the same constant feature vector which perfectly maximizes their similarity – resulting in a *collapsed* solution. The way to prevent this is by *contrasting* the positive pair by a set of negative pairs for which the model has to minimize the similarity. This prevents a collapsed solution of constant features as it opposes this objective. As such the set of negatives plays a vital role in contrastive learning and contributes to the structure of the resulting feature space. A large set of negatives can typically be obtained from different videos in the dataset; however, these often show different scenes, movements and actions than the query and are relatively easy to distinguish, therefore, often referred to as *easy* negatives. One way to enhance the set of negatives is by constructing specifically *hard* negatives that are designed to make the task challenging. *E.g.* the early work of (*van den Oord et al.*, 2018) uses patches of the same image in different locations as hard negatives. Similarly, the work of (*Han et al.*, 2019, 2020b) incorporate features at different spatio-temporal locations in the feature map as hard negatives for the future prediction task. The intra-contrastive loss in (*Yao et al.*, 2021) and (*Kuang et al.*, 2021) uses different frames of the same video as negatives, which forces the model to pay close attention to the variations between the frames. *Tao et al.* (2020) construct hard negatives by repeating a single frame or shuffling all frames of the query clip. Recently, *Dave et al.* (2022) proposed to extend the instance recognition loss with a "local-local" and a "global-local" contrastive loss in order to learn temporally distinct features. The "local-local" contrastive loss operates on temporally non-overlapping fixed lengths clips of the same video, while the "global-local" contrastive loss is applied on local clips and their corresponding timestamp in the feature map of the global video. For each of these two losses they consider clips/timestamps at the same temporal location as positives and at all other temporal locations as negatives.

Hard Positives. Hard negatives are only one way to form more challenging pairs. The counterpart to hard negatives are *hard positives*: here, the difficulty of the task arises from a more comprehensive set of (often several) positive samples, which are more difficult to identify than simple augmentation based positives. *Zhuang et al.* (2020) apply local aggregation to the video domain: Local aggregation (*Zhuang et al.*, 2019) augments the instance recognition loss by allowing similar data points to be grouped together in feature space. While *Zhuang et al.* (2020) find similar samples by clustering the feature space using k-means, *Tokmakov et al.* (2020) obtain the clusters in IDT feature space. Similarly, *Han et al.* (2020a) construct a set of *hard positives* by clustering a different feature space. Namely, they train two separate networks, one on RGB input the other on optical flow, and obtain the hard positives for the RGB model by taking the clusters in the feature space of the optical flow model,

and vice versa. As a result, two visually distinct videos with similar motion patterns will be embedded close. In contrast, *Xiao et al.* (2021a) aim to directly distill motion information: They also train two separate models on visual and motion input, respectively, each trained with a contrastive loss for instance recognition, and apply a third *motion-to-visual* contrastive loss where motion features are paired with the visual features. In a sense, cross-modal contrastive learning can be interpreted as hard positives as well: determining the same instance in a different modality is much more challenging than within modalities due to very distinct characteristics and typically requires reasoning at a higher level. Specifically, it prevents shortcuts via low-level cues. A very prominent example of this is CLIP (*Radford et al.*, 2021) in the image domain: They train their model on a large collection of image-text pairs using the InfoNCE loss to maximize the similarity between the image and its text description. The learned representations can be directly applied for zero-shot classification, where the pretrained text encoder can be repurposed to construct a classifier (by embedding the class names of the dataset). *Xu et al.* (2021) adapt their approach to the video domain. *Patrick et al.* (2021) study the effect of data transformations in a video-audio and a video-text representation learning setting. They consider transformations such as temporal shift, modality slicing, time reversal and spatial/aural augmentations and find that being invariant to all transformations in not the optimal solution. Instead, combining invariance to some transformations with distinctiveness to others is superior. *Zolfaghari et al.* (2021) enhance the cross-modal contrastive loss by taking *intra-modality* similarities into account as well: While the vanilla cross-modal contrastive loss only considers inter-modality negatives, they incorporate intra-modality negatives as well. Furthermore, they exclude highly related samples that are connected to many other samples and exclude them from the set of negatives to avoid false negatives.

2.1.4 Non-Contrastive Representation Learning

The success of contrastive learning, grounded in an instance recognition task that maximizes the similarity of two different samples showing the same instance, has inspired a new trend in self-supervised representation learning, termed *non-contrastive* learning: In a similar manner to contrastive learning, it aims to maximize the similarity of positive pairs; in contrast to contrastive learning however, there are no negative pairs. Naturally, this poses the question how a collapsed solution, that maps all samples to the same single feature vector, can be prevented. While contrastive methods rely on negative pairs to prevent collapse, non-contrastive methods need to turn to different mechanisms. *Grill et al.* (2020) pass two differently augmented versions of the same image through an online and a target network (each consisting of a backbone network followed by a projection head) and directly predict the representation of the target network from the representation of the online network with a prediction head. Their self-supervised loss directly maximizes the cosine similarity between target representation and online prediction. The online network is train end-to-end, while the target network is a slowly moving exponential average of the online network to which no gradients are backpropagated. They hypothesize that the updates of the target network's parameters are not in the gradient direction of the loss function and preventing a collapse that way. They experimentally find the moving average component to be crucial for performance and observe that an extreme case that instantly updates the target network to the online network results in collapse. *Chen and He* (2021) on the other hand find that this stop-gradient operation is sufficient for learning powerful representations. While it has been speculated by researchers that these methods avoid collapse via the batch statistics that are

leaked when applying BatchNorm (*Ioffe and Szegedy*, 2015) before computing the loss, *Richemond et al.* (2020) find that BYOL also works without BatchNorm and uses GroupNorm (*Wu and He*, 2018) instead. *Tian et al.* (2021b) investigate different components of non-contrastive self-supervised learning by analyzing the nonlinear learning dynamics. They find that the stop gradient operation on the target network and the prediction head on top of the online network are the essential ingredients to avoid collapse. *Caron et al.* (2021) propose a method with only a centering and sharpening of the teacher output to avoid collapsed solutions and do not depend on a predictor network. While these methods rely on multiple carefully designed components to avoid representational collapse, *Zbontar et al.* (2021) propose a conceptually simple method based on a principle of redundancy reduction to prevent collapsed solutions. They design a loss function that maximizes the similarity of feature vectors of two differently augmented images, while simultaneously minimizing the redundancy between different components of these vectors, measured by the cross-correlation. The above methods are mainly applied to representation learning on images. *Recasens et al.* (2021) apply a non-contrastive loss for video representation learning. More specifically, they design positive pairs from narrow and broad views, *i.e.* clips spanning a shorter or longer temporal context, and predict from narrow to broad view and vice versa, using a regression loss and the stop gradient trick similar to (*Grill et al.*, 2020; *Chen and He*, 2021).

2.2 Action Recognition

Action recognition is one of the most common video understanding tasks and has become the de facto evaluation downstream task for video representation learning. Moreover, most network architectures used to train self-supervised video representation models were initially designed for action recognition tasks. In this section, we will first review a selection of action recognition models and refer to (*Zhu et al.*, 2020) for a more thorough overview, before introducing the action recognition datasets used throughout this thesis.

2.2.1 From 2D to 3D Convolutional Neural Networks and Transformers

2D CNNs. With the success of convolutional neural networks (CNN) for image classification tasks, they have found their way into video understanding models as well. Early methods extended image-based 2D CNNs to video data, *e.g.* by feeding frames independently and averaging their predictions or concatenating the frames along the channel dimension (*Karpathy et al.*, 2014) or by adding a temporal aggregation model on top, essentially treating a video clip as a sequence of images. In the latter case, the 2D CNN is used to extract frame-wise features. To carry information across the frames, an aggregation model is put on top of these frame-wise features and taking in the information of all frames predicts an action class for the video. Common choices for such aggregation models are temporally recurrent networks, such as Recurrent Neural Networks (RNN) specifically Long Short Term Memory networks (LSTM) (*Hochreiter and Schmidhuber*, 1997), *e.g.* used in (*Donahue et al.*, 2015; *Yue-Hei Ng et al.*, 2015). In this case, the fusion of temporal information happens in the very last layers, where the content of the frames is already very abstracted, potentially losing crucial low-level motion information.

Two-Stream Models. One way to model such low-level motion is via optical flow. For example, Two-Stream Networks (*Simonyan and Zisserman*, 2014) take a single RGB frame and a set of stacked optical flow frames as input, train two separate 2D CNNs on top of each input and average their predictions. In this approach, the results of the two streams are combined at the very end, which referred to as late fusion, and prevents interaction between the two streams in intermediate layers. *Feichtenhofer et al.* (2016) investigate different fusion paradigms including how and where to perform fusion.

3D CNNs. A natural extension of 2D convolutions to video data simply treats the temporal dimension as an additional dimension analogous to the spatial dimensions by extending the 2D kernel to a 3D kernel (*Tran et al.*, 2015). This enables the model to capture low-level temporal information such as motion more effectively. A major breakthrough has been achieved by *Carreira and Zisserman* (2017) who adapt 2D image classification networks (specifically Inception-V1 (*Szegedy et al.*, 2015)) to 3D CNNs by “inflating" the 2D filters and kernels and equipping them with a temporal dimension. In order to reuse ImageNet (*Deng et al.*, 2009) pretrained weights to initialize the model, they inflate them in a similar manner, *i.e.* repeating the 2D filters along the temporal dimension and scaling them appropriately. They demonstrate that their model can be used in a two-stream setting as well in which one stream takes a set of RGB frames while another takes optical flow frames. Since then, advances of 2D CNN architectures in the image domain have inspired adapted 3D CNN versions. For example, *Hara et al.* (2018) simply replace the 2D convolutional filters of ResNets (*He et al.*, 2016) by 3D ones; the models we use in Chapter 4, 6 and 7 are based on 3D-ResNet18. *Tran et al.* (2018) and *Qiu et al.* (2017) further improve this architecture by factorizing the 3D kernels into separate 2D spatial and 1D temporal ones. *Xie et al.* (2018) mix 2D convolutions in early layers of the model with 3D convolutions in later layers; specifically, they factorize the 3D kernels similar to (*Tran et al.*, 2018) and (*Qiu et al.*, 2017). *Feichtenhofer et al.* (2019) proposes a model with a slow and a fast pathway: The slow pathway operates at a low frame rate to capture semantics while the fast pathway uses a high temporal resolution to capture quickly changing motion better. The above methods are designed to process a limited number of frames at a time (in many cases 16 or 32 consecutive frames) lacking the capacity to model longer-range temporal dependencies. Therefore, motivated by the observation that consecutive frames are highly redundant, *Wang et al.* (2019b) propose a global sparse frame sampling strategy that selects frames spread across the video. They divide the global video into a fixed number of segments and then randomly select a single frame within each segment, passes it through a 2D CNN and then aggregates them, *e.g.* via average pooling.

Transformer-based Networks. Inspired by the tremendous success of the Transformer model (*Vaswani et al.*, 2017) in the natural language processing domain, *Dosovitskiy et al.* (2021) propose a Vision Transformer (ViT) for image classification, which has since become widely adopted for image understanding tasks. Specifically, the ViT model utilizes the Transformer encoder model and feeds it a sequence of encoded tokens corresponding to image patches embedded using a linear projection and prepends a class token to the beginning of the sequence. The Transformer encoder model processes the sequence through several self-attention layers and outputs an encoded sequence of the same length as the input sequence; an MLP head is applied on the feature corresponding to the class token to predict classes. The Transformer model is designed to capture long-range contextual dependencies through the self-attention mechanism, a characteristic that makes them especially

appealing for long-range temporal modelling in video applications. Indeed, similar Transformer-based architectures have been proposed in the video domain as well: *Arnab et al.* (2021) propose a pure Transformer-based model for action recognition and investigate different options to partition the video into spatio-temporal tokens, including tokenizing each frame independently and concatenating the tokens of all frames and a tubelet embedding that divides the video into 3D tubes to better fuse spatio-temporal information during the tokenization step. They further explore several methods to factorize the model along the spatial and temporal dimension. Concurrently, *Bertasius et al.* (2021) propose a Transformer model that operates on a sequence of tokens from frame-level patches. They study different self-attention designs and find that a "divided attention", that applies spatial and temporal attention separately, works best. One drawback of the attention mechanism is its scalability as the computational complexity is quadratic in the length of the input sequence making it prohibitively expensive for high-resolution visual inputs. To alleviate these issues, *Liu et al.* (2021a) propose the Swin Transformer architecture that constructs a hierarchical feature maps by restricting the self-attention to non-overlapping local windows which are shifted between layers to allow cross-window connections. This reduces the computational complexity to linear runtime. *Liu et al.* (2021b) adapt the Swin Transformer to the video domain.

2.2.2 Datasets

Throughout the thesis, we conduct experiments on three commonly used datasets for action recognition: Kinetics-400 (*Kay et al.*, 2017), UCF101 (*Soomro et al.*, 2012) and HMDB51 (*Kuehne et al.*, 2011). We provide an overview of these datasets in Table 2.1 and discuss them in more detail below.

Dataset	Number of Classes	Number of Videos	Clips per Class	Clip Length
HMDB51 (*Kuehne et al.*, 2011)	51	6.766	min 102	3.1 sec
UCF101 (*Soomro et al.*, 2012)	101	13.320	min 101	7.2 sec
Kinetics-400 (*Kay et al.*, 2017)	400	247.218	min 400	10 sec

Table 2.1: **Statistics of Action Recognition Datasets.** We provide an overview of the statistics of three commonly used action recognition datasets.

HMDB51. The HMDB51 dataset (*Kuehne et al.*, 2011) was one of the largest datasets for action recognition at the time of its release. And although from today's perspective, with several large-scale datasets available, it is viewed as a smaller dataset, it is still one of the most popular datasets for evaluating video recognition models. The dataset consists of 51 action classes, each containing at least 102 clips. The action classes can be grouped into five different types: facial actions (*e.g.* smile, talk), facial actions with object interactions (*e.g.* eat, drink), body movements, body movements with object interactions and body movements with human interactions. The video clips are sourced from different internet sources, such as YouTube and Google videos, and movies, and are at least 1 second, on average 3.1 seconds long, with a frame rate of 30 FPS.

UCF101. The UCF101 dataset (*Soomro et al.*, 2012) is a mid-sized action recognition dataset with 101 action classes, each containing at least 101 video clips. The video clips are on average 7.2 seconds long with a frame rate of 25 FPS. In contrast to many previous datasets that consist of videos recorded in unrealistic, controlled settings, the UCF101 dataset consists of user-uploaded YouTube videos, which are recorded in the wild and can contain cluttered backgrounds and camera motions. The action classes in UCF101 are human-centered and can be broadly categorized into five different types: body motion, human-object and human-human interaction, playing instruments, and sports.

Kinetics-400. The Kinetics-400 dataset (*Kay et al.*, 2017) is a large scale dataset for action recognition, consisting of 400 action classes covering a broad range of human focused actions. Actions include single-person, person-object and person-person interactions and each action class contains at least 400 clips. The clips are about 10 seconds long with a frame rate of 30 FPS, *i.e.* clips cover around 300 frames each. The video clips are sourced from YouTube videos and the original dataset contains a total of 306.245 videos. Since the videos are collected from YouTube, Kinetics-400 is not a static dataset – videos can be deleted from YouTube and no longer be available, decreasing the size of the dataset over time. Our copy of the dataset consists of 234.584 training videos and 12.634 validation videos, comprising a total of 247.218 video clips.

2.3 Temporal Action Segmentation

In contrast to action recognition, which aims to assign a single action label to short trimmed videos, the task of temporal action segmentation is to temporally segment long untrimmed videos and classify each segment. In Chapter 7, we develop a Transformer-based model for temporal action segmentation, which can be applied both in the fully supervised and the timestamp supervised setting. Here, we review existing approaches in both categories. Since end-to-end training on the full untrimmed videos is too computationally expensive, the action segmentation task is typically decoupled into two steps: First, low-level spatio-temporal features are extracted from the raw frames and then, in a second step, a separate method for high-level temporal modelling is trained on top of the features to temporally segment the video.

2.3.1 Fully Supervised Action Segmentation

Early approaches are based on sliding window and non-maximum suppression (*Rohrbach et al.*, 2012; *Karaman et al.*, 2014). Other traditional approaches use hidden Markov Models (HMM) for high-level temporal modeling (*Kuehne et al.*, 2016; *Tang et al.*, 2012). *Lea et al.* (2016a) propose a spatio-temporal CNN to extract features and use a semi Markov Model on top for inference. To that end, they first train a 2D CNN to classify frames independently and in a second stage train a 1D temporal convolutional model on the frame-wise features of the 2D model. *Richard and Gall* (2016) use a language and length model to model the probability of action sequences and convert the frame-wise probabilities into action segments using dynamic programming.

More recent approaches are based on temporal convolutions to classify the video frames directly: *Lea et al.* (2017) propose two types of Temporal Convolutional Networks (TCNs) to capture long-range temporal dependencies. The first model consists of an encoder-decoder structure with several layers of temporal convolutions and pooling operations in the encoder to efficiently increase the

temporal receptive field and enable to model long-range dependencies. The decoder architecture is similar to the encoder but with temporal upsampling instead of pooling to map back to the original input sequence length, producing a sequence of frame-wise features. Finally, a linear classifier is applied to the features of each individual frame to predict its action class. The second model uses dilated convolutions instead of temporal pooling and upsampling to capture long-range dependencies. The dilation rate increases with each layer in a block to enable a large receptive field. *Lei and Todorovic* (2018) propose a Temporal Deformable Residual Network consisting of a residual stream that operates on the full temporal resolution and a pooling/unpooling stream that captures long-range dependencies via deformable temporal convolutions at multiple scales. While TCNs have been proven successful in practice, they still operate only on a low temporal resolution of the input frames. Furthermore, the temporal pooling operation may struggle to maintain fine-grained temporal information. To alleviate this problem, *Farha and Gall* (2019) propose multi-stage TCNs (MS-TCNs) that operate on the full input resolution. In each stage, they apply a set of dilated temporal convolutions with increasing dilation rate which generate an initial prediction in the first stage that is refined by the subsequent stages. After each stage, they apply a frame-wise classification loss and a smoothing loss to penalize over-segmentation errors. *Li et al.* (2020) extend this work in two ways: First, they propose dual dilated temporal convolutions in the first stage of the model to combine both a small and large receptive field in the early prediction stage. Second, they decouple the prediction generation and refinement stages of the model to customize the architecture to the respective tasks. *Wang et al.* (2020c) propose boundary-aware cascade networks based on the architecture of (*Farha and Gall*, 2019) with an adaptive temporal receptive field to dynamically respond to sudden label changes between frames. The early stages of their model focus on classifying simple frames, while later stages are aimed at more difficult ambiguous frames. Recently, *Yi et al.* (2021) combine multi-stage TCNs (*Farha and Gall*, 2019) with Transformer modules, where the receptive field in the self- and cross-attention layers is locally restricted to yield a hierarchical representation. More specifically, for each dilated temporal convolution in the MS-TCN architecture, they add an attention layer with instance normalization. The first stage of their model is the encoder, while the later stages are "decoders", which take the concatenated features of the encoder and the features of the previous stage as input to the cross-attention layer. While we use a similar encoder as ASFormer for our model in Chapter 7, our decoder is very different and follows the auto-regressive prediction paradigm of the traditional Transformer model (*Vaswani et al.*, 2017); more details of our architecture can be found in Section 3.3.3. The above methods solve the action segmentation task by predicting an action class for each frame. Unfortunately, this approach is prone to over-segmentation and requires refinement modules and smoothing or post-processing and expensive inference. *Ishikawa et al.* (2021) addresses this issue by introducing a boundary regression branch to detect action boundaries, which are used during inference to refine the segmentation. *Huang et al.* (2020) propose a Graph-based Temporal Reasoning Module that can be built on top of existing methods to refine their predicted segmentations. In contrast, our method in Chapter 7 aims to predict the high-level sequence of action segments directly.

2.3.2 Weakly Supervised Action Segmentation

Training action segmentation models in a fully supervised setting requires fine-grained temporal annotations, *i.e.* precise localization when one action ends and another begins. To avoid these costly

frame-wise annotations, many methods have been proposed that rely on a weaker form of supervision, such as transcript supervision (*Bojanowski et al.*, 2014): Here, only the ordered sequence of actions that occur in the video are given, but not their temporal location. *Huang et al.* (2016) extend the connectionist temporal classification framework, originally introduced for speech recognition, to videos to efficiently evaluate all possible frame-to-action alignments. Since these can include visually inconsistent alignments, they incorporate a frame-to-frame similarity into the framework to enforce consistency. *Ding and Xu* (2018) propose a new TCN variant in combination with an iterative soft boundary assignment strategy to generate frame-wise pseudo-labels from transcripts during training. To that end, they map a transcript to a sequence of frame-wise labels assuming uniform segment lengths and construct soft boundaries between segments by interpolating the probabilities of the two actions. The transcripts are then iteratively refined based on the predictions of the current model with an insertion strategy; namely, an action label is repeated if its probability at the action boundary exceeds that of the neighboring action by a given threshold. *Richard et al.* (2018) generate frame-wise pseudo-labels with the Viterbi algorithm: The Viterbi algorithm can be used to compute the global optimal segment lengths given frame-wise class probabilities and a transcript and is traditionally used to convert potentially noisy frame-wise predictions into smooth segment predictions. *Richard et al.* (2018) take advantage of this characteristic and determine the optimal segmentation of the current predictions and the ground truth transcript in each training iteration. They then use the obtained segmentation as pseudo ground truth in a frame-wise classification loss. *Li et al.* (2019) extend this work by discriminating between all valid and invalid segmentations of a video. *Souri et al.* (2021b) use a two-branch neural network with a frame classification branch and a segment generation branch, which predict two redundant segmentations – one at a frame-level, the other at a segment-level. The two different representations types aim to predict the same segmentation and can be used to guide each others training by enforcing them to be consistent via a mutual consistency loss. In addition, the action prediction of the segment generation branch can be supervised by the ground truth transcript. More recent approaches exploit Dynamic Time Warping for the weakly supervised action segmentation task: Dynamic Time Warping can be used to align two sequences using a dynamic programming procedure. *Chang et al.* (2019) use Dynamic Time Warping to align the video frames to the video transcript and obtain a differentiable loss with a continuous relaxation of the min-operator in the dynamic programming procedure. *Chang et al.* (2021) on the other hand align video sequences with an ordered sequence of action prototypes representing the transcript. The learned discriminative action prototypes can then be used during inference to align frames to transcripts. While transcript supervision reduces the annotation cost significantly, it has been observed that the performance suffers in practice. As an alternative, timestamp supervision (*Li et al.*, 2021c) has been proposed, where for each action segment a single frame is annotated. The annotation cost for such timestamps is comparable to transcript annotations (*Li et al.*, 2021c) but provides a stronger level of supervision as it gives information about the rough location of the segments. In Chapter 7, we evaluate our method on this form of supervision as well.

2.3.3 Datasets

In Chapter 7, we perform experiments on three commonly used datasets for action segmentation, namely Breakfast (*Kuehne et al.*, 2014), 50Salads (*Stein and McKenna*, 2013) and GTEA (*Fathi et al.*, 2011); an overview of the datasets statistics can be found in Table 2.2. Among these datasets

Breakfast is the largest dataset available for the action segmentation task to date. In contrast, 50Salads and GTEA contain much fewer videos and classes, but more segments per video. Furthermore, 50Salads contains the longest videos, spanning an order of magnitude more frames on average.

Dataset	Classes	Videos	Min Length	Max Length	Mean Length	Min # of Segments	Max # of Segments
GTEA (*Fathi et al.*, 2011)	11	28	634	2009	1115	21	44
50Salads (*Stein and McKenna*, 2013)	17	50	7555	18143	11552	15	26
Breakfast (*Kuehne et al.*, 2014)	48	1712	130	9741	2097	2	25

Table 2.2: **Statistics of Action Segmentation Datasets.** We provide an overview of the statistics of three commonly used datasets for temporal action segmentation, including the number of classes and videos in the dataset, the minimum/maximum/mean length of the videos, and the minimum/maximum number of segments in a video.

GTEA. The Georgia Tech Egocentric Activities dataset (*Fathi et al.*, 2011) consists of 28 videos recorded by 4 subjects using a GoPro camera. The camera is mounted on top of the head of each subject and records the area in front of the subject's eyes, *i.e.* from an egocentric point of view. The frames are extracted at a 15 fps rate. The videos are on average 74.3 seconds long, or 1115 frames, and contain 21 to 44 action segments. The videos show seven different activities related to preparing food, *e.g.* "peanut butter sandwich", each made up of several sub-actions, such as "spoon" and "spread". In total, the dataset consists of 10 action classes and one background class.

50Salads. The 50Salads dataset (*Stein and McKenna*, 2013) consists of videos of food preparation as well, it shows 25 actors preparing two mixed salads each; in total the dataset consists of 50 videos. The videos are recorded using an RGB-D camera from a top-view perspective at a frame rate of 30 fps. In comparison with the GTEA dataset, the videos of 50Salads are much longer: they span 385 seconds or 11552 frames on average. The dataset consists of 17 action classes and each video contains 15 to 26 action segments.

Breakfast. The Breakfast dataset (*Kuehne et al.*, 2014) is one of the largest available action segmentation datasets, consisting of 1712 videos. A total of 52 participants perform 10 breakfast cooking activities in multiple different kitchens. The video were recorded from a third person view using multiple cameras at a frame rate of 15 fps and are on average 139.8 seconds or 2097 frames long. Overall, the dataset consists of 48 action classes; each video contains 2 to 25 action segments.

CHAPTER 3

Preliminaries

In this chapter, we formally define the video representation learning task and discuss some common evaluation strategies. To that end, we first introduce the basic notation and spell out the merits of deep learning for video representation learning. In this thesis, we are interested in *self-supervised* video representation learning via contrastive learning. As contrastive learning lays the foundation of our methods in Chapters 4, 5 and 6, we will introduce the most important components of contrastive learning. Equipped with the necessary tools for *learning* unsupervised video representations, the question remains how to *evaluate* them properly. In contrast to standard computer vision tasks, a "ground truth representation" that we can compare to is not available; in fact, it may be difficult to determine what an optimal representation should look like and may even depend on the target task at hand. Therefore, downstream tasks have become the de facto evaluation of learned representation: the quality of a representation is rated based on its performance on a target task. A wide range of video understanding tasks can be used for this purpose; here, we review three downstream tasks that we use throughout the thesis. This also includes the temporal action segmentation task, for which we develop a method in Chapter 7. Finally, we introduce the neural network architectures used throughout the thesis, including deep residual networks, gated recurrent units and the Transformer model.

Contents

3.1 Video Representation Learning

At an abstract level, (video) representation learning can be defined in different ways, *e.g.* one could argue that it should identify and disentangle the underlying factors of variation in our data (*Bengio*

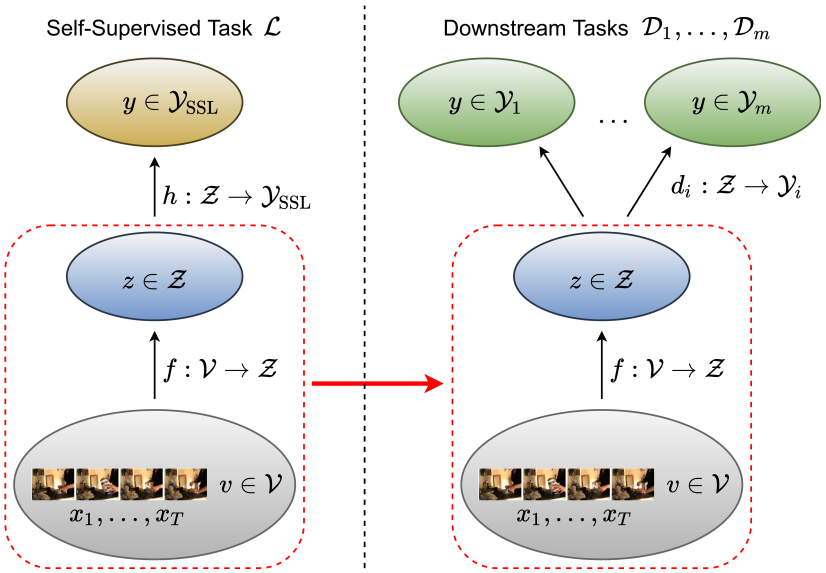


Figure 3.1: **The Self-Supervised Video Representation Learning Task.** Self-supervised video representation learning aims to learn a model f to extract high-level abstract concepts from videos $v \in \mathcal{V}$, represented by features $z \in \mathcal{Z}$ (blue feature space), without using any labels. This is typically done via self-supervised pretraining (left), where the model f, also called backbone, together with a self-supervised head h is trained to solve a self-supervised task $\mathcal{L}$. The head h maps the features $z \in \mathcal{Z}$ to the self-supervised target space $\mathcal{Y}_{\text{SSL}}$ where the loss is applied and is typically discarded after the self-supervised pretraining. An ideal representation benefits a diverse set of downstream tasks $\mathcal{D}_1, \ldots, \mathcal{D}_m$ (right). Here, the self-supervised pretrained model f (left) serves *e.g.* as a feature extractor or initialization for the downstream tasks (right) and is equipped with a classifier d_i that maps the features $z \in \mathcal{Z}$ to the targets $y \in \mathcal{Y}_i$ of the downstream task $\mathcal{D}_i$.

et al., 2013); however, the ground truth factors of variation for such representations is rarely available. Therefore, we define video representation learning from a more practical point of view: A useful representation should give us some benefit on a target task – ideally, multiple target tasks. This definition also simplifies the assessment of the quality of learned representations.

While early work constructs hand crafted features, *e.g.* improved dense trajectories (IDT) (*Wang and Schmid*, 2013) and space time interest points (*Laptev and Lindeberg*, 2003), modern approaches are almost exclusively based on deep learning. It allows the model to discover the "optimal" features, *i.e.* the features that are well suited to solve the task, on its own. While supervised learning is still a popular approach to extract meaningful representations, interest in unsupervised methods has increased over the years.

3.1.1 Problem Formulation and Notation

In the following, we will introduce some basic notation and formalize the video representation learning task. At a fundamental level, videos make up the input space of all tasks and problems we consider in this thesis; therefore, we denote the space of all videos by $\mathcal{V}$. Each video $v \in \mathcal{V}$ consists of a sequence of T_v frames $v = (x_1, \ldots, x_{T_v})$, where $x_t \in \mathbb{R}^{3 \times H \times W}$ is the RGB frame of height H

and width W at time t. From a high-level perspective, the goal of video representations is to define a function $f : \mathcal{V} \to \mathcal{Z}$ that maps videos $v \in \mathcal{V}$ to an abstract feature space $\mathcal{Z}$. The features $z \in \mathcal{Z}$ should accurately represent high-level abstractions of the video content.

But how should the ideal feature space $\mathcal{Z}$ look like? One way to define what a good representation should look like is via some desired properties of the feature space, *e.g.* it should identify and disentangle the underlying factors of variation in the data (*Bengio et al.*, 2013). However, this perspective poses some challenges in practice: How do we train a model to achieve this disentanglement when the ground truth factors of variation are unknown? And even if we achieve disentangled representations will they be useful for downstream tasks? A recent study by *Locatello et al.* (2019) challenges this ideal.

From a more practical point of view, the features $z \in \mathcal{Z}$ should – if they represent the video perfectly – enable or at least facilitate the application to various downstream tasks $\mathcal{D}_1, \ldots, \mathcal{D}_m$, some of which we discuss in Section 3.2. The downstream tasks $\mathcal{D}_i$ typically define pairs (v, y) of inputs $v \in \mathcal{V}$ and their corresponding targets $y \in \mathcal{Y}_i$ (*e.g.* action labels). In this definition, an ideal feature space $\mathcal{Z}$ is one that enables a classifier $d_i : \mathcal{Z} \to \mathcal{Y}_i$ to solve the task $\mathcal{D}_i$ given the features $z \in \mathcal{Z}$. In many cases, the classifier d_i is a linear layer, which assumes that the learned features are linearly separable. This definition also simplifies the comparison of representations: a representation is deemed to be superior if it achieves better performance on a downstream task.

While early approaches for video representations hand engineered the function f, modern approaches define an unsupervised training objective to *learn* representations. The unsupervised task can be designed such that the learned representation expresses many generic priors, *i.e.* priors that are not task-specific but potentially useful for a target task (*Bengio et al.*, 2013), *e.g.* smoothness of f, temporal coherence, and sparse representations to list a few. Essentially, unsupervised representation learning boils down to learning a prior of the data $\mathbb{P}(\mathcal{V})$. The hypothesis is that the underlying factors of $\mathcal{V}$ help to model the targets $\mathcal{Y}_i$; thus, representations that are useful for $\mathbb{P}(\mathcal{V})$ will be useful for $\mathbb{P}(\mathcal{Y}_i|\mathcal{V})$ as well (*Bengio et al.*, 2013). However, not all underlying factors in the data distribution will be equally useful for all potential downstream tasks. In fact, the value of an unsupervised task may depend on how well it is aligned with the target task. Generally, the goal is to define an unsupervised task that aligns well with as many downstream tasks as possible.

3.1.2 Deep Learning for Representation Learning

Deep learning provides an especially convenient tool for video representation learning. Deep neural networks tend to naturally learn a *hierarchy* of features: early layers in the model typically capture low-level cues, while later layers usually capture more abstract concepts composed of less abstract ones from previous layers. This is especially fitting for video data as videos are inherently hierarchical by nature: a high-level action is typically made up of several sub-actions, *e.g.* different steps in preparing a meal, and those sub-actions in turn are again made up of several atomic actions and low-level movements, etc. Similarly, the feature space $\mathcal{Z}$ may have different levels, from fine-granular localized framewise features to a single feature vector representing the video at a global scale and various levels in between, *e.g.* a sequence features representing a sub-sequence of the video at a time. In principle, we can apply unsupervised losses at various layers in the model to guide the representation learning process at different stages. In this thesis, we take the representation at the penultimate layer, *i.e.* before the linear classification head, since we are aiming for high-level abstract features of

videos and allow the model to figure out the optimal representations in early layers on its own.

Deep neural networks are an umbrella term encompassing a larger family of models that map input to output through consecutive *layers* of differentiable functions – in many cases a linear transformation followed by a non-linear activation function. Each layer $f_{\theta_i}^{(i)}$ is parameterized by a set of weights θ_i; together $\theta = \{\theta_1, \ldots, \theta_l\}$ make up the weights of the full model

$$f_\theta = f_{\theta_l}^{(l)} \circ f_{\theta_{l-1}}^{(l-1)} \circ \cdots \circ f_{\theta_1}^{(1)}. \tag{3.1}$$

Neural networks are universal function approximators (*Hornik et al.*, 1989): they can approximate any continuous function arbitrarily well on a compact subset (assuming arbitrary width of the model). In other words, they can represent a very wide variety of different functions when the weights θ are chosen properly. Consequently, they have the potential to solve many complex tasks (including self-supervised tasks), which are typically expressed through a loss function $\mathcal{L}$.

In self-supervised learning, the challenge lies within designing $\mathcal{L}$ such that the task is sufficiently challenging for the network to solve and forces the model to explore the structure of the data thoroughly. Once we have defined a loss function $\mathcal{L}$, we train the model to minimize the expected loss over all possible videos. Commonly, a mapping $h : \mathcal{Z} \to \mathcal{Y}_{\text{SSL}}$ is employed that maps the output of f_θ to the space $\mathcal{Y}_{\text{SSL}}$, where the self-supervised loss $\mathcal{L}$ is applied. Here, h is the self-supervised task head and f_θ is the backbone. Note that h contains learnable parameters as well; for ease of notation we drop them in the equations below. We optimize the parameters $\theta \in \Theta$, which define the behavior of our model f_θ:

$$\arg\min_{\theta \in \Theta} \mathbb{E}_{\mathcal{V}} \left[\mathcal{L}(h(f_\theta(v))) \right]. \tag{3.2}$$

In practice, we do not have access to the true sample space $\mathcal{V}$ of all possible videos and instead approximate it by sampling a large number of videos $\{v_1, \ldots, v_n\}$, $v_i \in \mathcal{V}$, and the optimization problem boils down to

$$\arg\min_{\theta \in \Theta} \sum_{i=1}^{n} \mathcal{L}(h(f_\theta(v_i))). \tag{3.3}$$

We can optimize Eq. (3.3) with stochastic gradient descent (SGD), which approximates the gradient with respect to θ on a minibatch of samples.

Ultimately, we are not simply interested in minimizing $\mathcal{L}$ on all available samples $\{v_1, \ldots, v_n\}$, but rather we are aiming for *generalization* – a good performance of f_θ on *new* samples $v \in \mathcal{V}$ – which we assess by evaluating on held-out samples. Specifically, we typically divide the available samples into a training set for optimization, a validation set for hyper-parameter selection and a test set to test the generalization ability of the learned model.

3.1.3 Contrastive Learning

There is a large body of literature on how to design an unsupervised loss function $\mathcal{L}$; we reviewed a selection of self-supervised tasks for video representation learning in Section 2.1. In recent years, *contrastive learning* has established a particularly promising direction for this purpose. In this section, we will review the basics of contrastive learning in more detail, as they form the foundation of the methods in Chapters 4, 5 and 6.

Positive and Negative Pairs. Contrastive learning is intimately tied to the definition of similar and dissimilar pairs of samples: it aims to learn a feature space in which similar pairs – referred to as *positive pairs* – are embedded close to each other, while dissimilar *negative* pairs are pushed apart. As such, these sets of positive and negative pairs have a major impact on the structure of the feature space and constitute one of the most important components of contrastive learning. A widely adopted approach for defining positive pairs without using any labels is *instance recognition*. Here, we construct two different *views* of the same data that maintain the semantics of the original data. For example, we can apply two different augmentations, such as random crops and color transformations, to an image, which alter the appearance of objects but preserves the content of the image for the most part, essentially allowing our model to *view* the same data under different conditions.

Given a single sample q, which we refer to as *query*, we define a set of positives $p \in \mathcal{P}$ that are related to q and a set of negatives $n \in \mathcal{N}$, *i.e.* unrelated samples. We denote the respective features by $z_q = f_\theta(q)$, $z_p = f_\theta(p)$ and $z_n = f_\theta(n)$.

Cosine Similarity. Our aim is to maximize the similarity between the features of positive pairs (z_q, z_p), $p \in \mathcal{P}$, and minimize the similarity of negative feature pairs (z_q, z_n), $n \in \mathcal{N}$. To that end, we first define a similarity measure; we use the cosine similarity throughout the thesis:

$$\mathrm{sim}(z_1, z_2) = \frac{z_1^T z_2}{\|z_1\| \|z_2\|}. \tag{3.4}$$

A very popular variation applies another learnable transformation h (in many cases an MLP head) and scales the cosine similarity with a temperature parameter τ (*Chen et al.*, 2020b); for simplicity we denote these adaptions by

$$\mathrm{sim}_{\tau,h}(z_1, z_2) = \frac{1}{\tau} \frac{h(z_1)^T h(z_2)}{\|h(z_1)\| \|h(z_2)\|}. \tag{3.5}$$

If we set $\tau = 1$ and use the identity function $h = \mathrm{id}$, Eq. (3.5) is equal to Eq. (3.4). We generally assume the form in Eq. (3.5) and occasionally drop the subscripts τ and h to simplify the notation when they are clear from the context.

InfoNCE Loss. While many contrastive losses, which aim to maximize the similarity of positive pairs and minimize the similarity of negative pairs, have been proposed in the field of metric learning, the InfoNCE loss has recently gained a lot of popularity. In its most basic form, $\mathcal{P}$ contains only a single positive p and the InfoNCE loss reads

$$\mathcal{L}_{\text{InfoNCE}} = -\sum_q \log \frac{\exp\left(\mathrm{sim}_{\tau,h}(z_q, z_p)\right)}{\exp\left(\mathrm{sim}_{\tau,h}(z_q, z_p)\right) + \sum_{n \in \mathcal{N}} \exp\left(\mathrm{sim}_{\tau,h}(z_q, z_n)\right)}. \tag{3.6}$$

Since the $\log$ function is monotone, minimizing Eq. (3.6) breaks down to maximizing the expression inside the $\log$, which is achieved when $\mathrm{sim}_{\tau,h}(z_q, z_p)$ is maximized and $\mathrm{sim}_{\tau,h}(z_q, z_n)$ is minimized for $n \in \mathcal{N}$, and is precisely what we are aiming for in contrastive learning.

The basic InfoNCE loss above assumes that a single positive is given, which may be overly restrictive when we can easily obtain multiple positive samples for the query q. For example, we can treat samples of the same class also as positives (*Khosla et al.*, 2020) or we can sample several short

clips from the same video (*Feichtenhofer et al.*, 2021). One straightforward way to include several positives into the contrastive loss is to apply Eq. (3.6) to each positive, *i.e.* take the sum over the positives *outside* of the log:

$$\mathcal{L}^{\text{out}}_{\text{InfoNCE}} = -\sum_{q}\sum_{p\in\mathcal{P}} \log \frac{\exp\left(\text{sim}_{\tau,h}(z_q, z_p)\right)}{\exp\left(\text{sim}_{\tau,h}(z_q, z_p)\right) + \sum_{n\in\mathcal{N}} \exp\left(\text{sim}_{\tau,h}(z_q, z_n)\right)}. \tag{3.7}$$

This loss enforces all positive pairs to be similar, which is useful when we are confident that all positives are true positives. However, in some cases we may be facing a more noisy set of positives – *e.g.* sampling temporally distant clips may show drastically different content due to changes in the video – and forcing all of them to be similar can reduce the representation quality. To alleviate this issue, we can alternatively compute the sum *inside* the log:

$$\mathcal{L}^{\text{in}}_{\text{InfoNCE}} = -\sum_{q} \log \frac{\sum_{p\in\mathcal{P}} \exp\left(\text{sim}_{\tau,h}(z_q, z_p)\right)}{\sum_{p\in\mathcal{P}} \exp\left(\text{sim}_{\tau,h}(z_q, z_p)\right) + \sum_{n\in\mathcal{N}} \exp\left(\text{sim}_{\tau,h}(z_q, z_n)\right)}. \tag{3.8}$$

In contrast to Eq. (3.7), Eq. (3.8) does not force all positives to be similar; essentially, this loss is satisfied as long as a sufficiently large similarity is set for at least one positive.

Connection to Mutual Information. The InfoNCE loss has been associated with mutual information early on: *van den Oord et al.* (2018) derive the InfoNCE loss as a lower bound of mutual information, suggesting the success of the proposed loss function originates from maximizing the mutual information between the two variables. Mutual information measures the amount of information obtained about one random variable given the other and is formally defined as the KL-divergence between the joint density $p(x, y)$ and the product of the marginals $p(x)$ and $p(y)$:

$$I(X;Y) = D_{\text{KL}}\left(p(x,y) \| p(x)p(y)\right). \tag{3.9}$$

The lower bound derived in (*van den Oord et al.*, 2018) relates the InfoNCE loss to mutual information in the following way:

$$I(X;Y) \geqslant \log(N) + \frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp(f(x_i, y_i)}{\sum_{j=1}^{N} \exp(f(x_i, y_j)} = \log(N) - \mathcal{L}_{\text{InfoNCE}}. \tag{3.10}$$

The interpretation is that by minimizing $\mathcal{L}_{\text{InfoNCE}}$ we effectively maximize a lower bound of the mutual information and therefore indirectly the mutual information itself. While *Poole et al.* (2019) prove that the InfoNCE loss is indeed a lower bound on mutual information if the negative samples are drawn from the true marginal distribution, *Tschannen et al.* (2020b) point out that this assumption is often violated in practice when *hard negatives* are constructed explicitly with the intention to improve the learned representations. Furthermore, they find empirically that higher capacity critics, which achieve a tighter bound on the mutual information, do not necessarily transfer to better downstream performance and challenge the conception that mutual information maximization alone is a useful objective for representation learning. This is in line with observations from representation learning in the image domain, where a better performance on the pretext task does not indicate better performance on downstream tasks (*Noroozi and Favaro*, 2016; *Kolesnikov et al.*, 2019).

Invariance of Learned Representations. Contrastive representation learning is closely tied to the concept of *invariances* (*Purushwalkam and Gupta*, 2020), which are induced by the transformations that we apply to obtain our positive pairs. As the model aims to learn similar representations of positive pairs, it is prone to focus on the information that is shared between them and will tend to omit cue that are distinct to each individual view. We say our model f is invariant to a transformation t if $f(t(v)) \approx f(v)$. In instance recognition – the most widely adopted approach of modern contrastive methods – two views of a sample $v \in \mathcal{V}$ are generated by applying two independent augmentations $t_1, t_2 \in \mathcal{T}$ which provide the positive pair. Training InfoNCE on these pairs entails $f(t_1(v)) \approx f(t_2(v))$. In many cases, $\mathcal{T}$ includes the identity or invertible transformations making our model invariant to the set of augmentations in $\mathcal{T}$. Besides invariance to the augmentations themselves, they can also induce invariance to other transformations: For example, *Purushwalkam and Gupta* (2020) find that the heavy crop augmentation, which is commonly used in the image domain, leads to occlusion invariant representations. The aggressive cropping creates pairs of image patches that show different parts of an object, imitating a form of occlusion. Another example comes from natural transformation that objects go through over time, such as changing viewpoints and deformations. Using temporal shift as augmentation on videos aims to be invariant to such types of transformations.

3.2 Video Understanding Tasks

In the previous section, we have layed the groundwork of video representation learning. An ideal representation should extract high-level information describing the contents of the input data accurately. However, there typically is no ground-truth representation that we can compare the learned model with; in fact, the optimal representation may depend on the target task at hand. Therefore, the most common quantitative evaluation of video representations is performed via *downstream tasks*, *i.e.* different video understanding tasks of interest. Here, our assumption is that improved representations, which represent videos more thoroughly, encompass useful features for the downstream task and consequently lead to a better performance. In the following, we will go over the three video understanding tasks considered in this thesis, starting with the most common and widely adopted action recognition downstream task in Section 3.2.1. Another popular evaluation is a video retrieval task, which we describe in Section 3.2.2. Both of these tasks assume a set of short trimmed videos, specifically on datasets which are inherently biased towards static features. Therefore, to evaluate the learned representations more thoroughly we consider temporal action segmentation as another downstream task in Section 3.2.3.

3.2.1 Action Recognition

Action recognition is one of the most prevalent video understanding tasks and has become the de facto evaluation of video representations. Most commonly, it involves short trimmed videos showing a single action being performed and the objective is to assign each video a single action label of a pre-defined set of possible actions $\mathcal{C} = \{c_1, \ldots, c_j\}$.

Finetuning Evaluation. Formally, the task is to train a model $m : \mathcal{V} \rightarrow \mathcal{C}$ that maps a video $v \in \mathcal{V}$ to its corresponding action class $c \in \mathcal{C}$. In the context of video representation evaluation, the classi-

fication model m is composed of the pretrained feature extractor $f_\theta : \mathcal{V} \to \mathcal{Z}$, also called *backbone*, that was obtained through the self-supervised learning stage, followed by a subsequent classification head $d_{\text{action}} : \mathcal{Z} \to \mathcal{C}$. The classification head d_{action} can be a linear layer or an MLP (we use a linear layer throughout the thesis) and is parameterized by ϑ. The full model $m = d_{\text{action}} \circ f_\theta$ is consequently parameterized by (ϑ, θ), the weights of the classification head and the pretrained weights of the backbone, and trained end-to-end on the classification task in a fully supervised setting:

$$(\vartheta^*, \theta^*) = \underset{(\vartheta,\theta)}{\arg\min}\, \mathbb{E}\left(\log p(m(v) = c)\right). \tag{3.11}$$

The above evaluation paradigm is referred to as *finetuning*: The pretrained model weights θ serve as an initialization θ_{init} to the backbone model. While this establishes a practical form of video representation evaluations – afterall this is what we would do in a real world application to boost performance as much as possible – it is a rather uncontrolled evaluation and prone to overfitting on small datasets.

Linear Evaluation. Alternatively, we can evaluate video representations under a linear evaluation protocol (often referred to as *linear probe*): Here, f_θ serves as a frozen feature extractor and only the classification head d_{action} is trained:

$$\vartheta^* = \underset{\vartheta}{\arg\min}\, \mathbb{E}\left(\log p(m(v) = c)\right). \tag{3.12}$$

This paradigm evaluates the linear separability of the learned representation with respect to the action classes. A drawback of this approach can be that it prevents early layers to adapt to different input distributions when evaluating on datasets different from the pretraining dataset.

Layer-wise Evaluation. A trade-off between the two evaluation schemes above can be to freeze the pretrained model only up to some layer $f^{(k)}$ and training the remaining layers and the classification head from scratch:

$$(\vartheta^*, \theta^*_{k+1}, \ldots, \theta^*_l) = \underset{(\vartheta,\theta_{k+1},\ldots,\theta_l)}{\arg\min}\, \mathbb{E}\left(\log p(m(v) = c)\right). \tag{3.13}$$

This evaluation was primarily proposed in the image domain (*Noroozi and Favaro*, 2016) and allows for a more comprehensive evaluation which gives better insights into the learned representations at different layers of the backbone. It is well known that deep neural networks learn a hierarchy of features, in which early layers extract more low-level local features which tend to be general-purpose features, while later layers capture more abstract concepts that enable the model to solve the training task (*Yosinski et al.*, 2014). Thus, a higher performance in early layers indicates better low-level general-purpose features, while the later layers reveal how well the self-supervised pretraining task is aligned with the downstream target task.

3.2.2 Video Retrieval

The parametric evaluation in the previous section comes with several drawbacks: The exact choice of the framework used for finetuning influences the final accuracies substantially; an apples-to-apples

comparison between different methods is impossible. Although, the linear probe evaluation mitigates this problem to some extend, it poses strong assumptions on what the ideal representation space should look like, *i.e.* linear separability with respect to the action classes, even if non-linear separability can be sufficient in practice. On top of this, and perhaps surprisingly, the training regime used for training the linear layer influences the classification accuracy as well (*Kolesnikov et al.*, 2019). For this reason, a non-parametric nearest-neighbor video retrieval evaluation has been adopted in the literature, typically performed on the same datasets that are used in the action recognition downstream tasks.

In this evaluation, the pretrained backcone serves as a feature extractor and is kept fixed. For a given dataset $\mathcal{D} = \{(v_1, y_1), \dots, (v_n, y_n)\}$ of video and action label pairs, we first extract features for all videos in the dataset, $\{(z_1, y_1), \dots, (z_n, y_n)\}$. We divide the dataset into a training set and a test set $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$ and denote the training and test videos by $\mathcal{V}_{\text{train}}$ and $\mathcal{V}_{\text{test}}$, respectively. Then, we compute the Recall@k (R@k) in the following way. For each video $v_{\text{test}} \in \mathcal{V}_{\text{test}}$ in the test set we retrieve the top k nearest neighbors:

$$\mathcal{N}^k_{v_{\text{test}}} = \underset{\mathcal{V} \subseteq \mathcal{V}_{\text{train}} : |\mathcal{V}| = k}{\arg\max} \sum_{v_i \in \mathcal{V}} \text{sim}(v_{\text{test}}, v_i), \tag{3.14}$$

and count a correct retrieval if at least one video in $\mathcal{N}^k_{v_{\text{test}}}$ is of the same class

$$\text{success}(v_{\text{test}}, k) = \begin{cases} 1 & |\{v_i \in \mathcal{N}^k_{v_{\text{test}}} : c(v_i) = c(v_{\text{test}})\}| \geqslant 1 \\ 0 & \text{else} \end{cases}, \tag{3.15}$$

where $c(v_i) = y_i$ determines the label $y_i \in \mathcal{C}$ of v_i. If the retrieval was successful (*i.e.* $\text{success}(v_{\text{test}}, k) = 1$) we consider the video v_{test} correctly classified. The R@k is then computed as

$$\text{R@}k = \frac{1}{|\mathcal{V}_{\text{test}}|} \sum_{v_{\text{test}} \in \mathcal{V}_{\text{test}}} \text{success}(v_{\text{test}}, k). \tag{3.16}$$

Note that R@k does not measure the precision of the retrieved videos, *i.e. how many* videos in $\mathcal{N}^k_{v_{\text{test}}}$ are of the same class. Therefore, we supplement the reported R@k values with precision-recall-curves throughout the thesis. To that end, we compute precision and recall for all values of k and plot the resulting curves. In this setting, precision and recall are calculated as it is the standard approach in retrieval – precision is the fraction of relevant instances among the retrieved instances, while recall is the fraction of relevant instances that were retrieved – averaged over the test set:

$$\text{precision@}k = \frac{1}{|\mathcal{V}_{\text{test}}|} \sum_{v_{\text{test}} \in \mathcal{V}_{\text{test}}} \frac{|\{v_i \in \mathcal{N}^k_{v_{\text{test}}} : c(v_i) = c(v_{\text{test}})\}|}{k}, \tag{3.17}$$

$$\text{recall@}k = \frac{1}{|\mathcal{V}_{\text{test}}|} \sum_{v_{\text{test}} \in \mathcal{V}_{\text{test}}} \frac{|\{v_i \in \mathcal{N}^k_{v_{\text{test}}} : c(v_i) = c(v_{\text{test}})\}|}{|\{v_i \in \mathcal{V}_{\text{train}} : c(v_i) = c(v_{\text{test}})\}|}. \tag{3.18}$$

3.2.3 Temporal Action Segmentation

Merely evaluating video representations on action recognition datasets does not assess the quality of the learned representations very thoroughly: the commonly used datasets are inherently biased

towards static features (*Li et al.*, 2018) and a sparse, global sampling strategy (*Wang et al.*, 2019b) works sufficiently well – even in extreme cases when only a single frame is used. Temporal information seems to be less important for these datasets and therefore does not provide a reasonable approach to evaluate temporal features in video representations. Therefore, in Chapter 6, we propose to extend the standard evaluation protocol via action recognition with a temporal action segmentation downstream task.

Action segmentation differs from action recognition is various ways: First, it is typically performed on long untrimmed videos opposed to pre-segmented short ones in action recognition. Second, it has a very different label space: In contrast to action recognition, where we have a single global action label per video, temporal action segmentation aims to predict a sequence of actions and their temporal locations in the video. Namely, these are dense annotations, *i.e.* we have a single action label *per frame*. Together this makes action segmentation an especially challenging task: Untrimmed videos tend to share many visual cues over time; they often show a similar background and actors and objects are visible throughout the video. Different actions within a single video are often distinguishable only via subtle variations. Furthermore, the pretrained backbone only serves as a frozen feature extractor here, allowing us to better evaluate its ability to represent such small temporal variations. We extract framewise features while keeping the input resolution of the pretrained model intact, and then train a separate action segmentation model on top of the framewise features. More specifically, the models that we use throughout this thesis are 3D-CNNs, which take a sequence of L frames as input at a time. To compute the features for video frame x_t at time t, we take a temporal window around it $(x_{t-\frac{L}{2}}, \dots x_{t+\frac{L}{2}})$ and feed it to the backbone f_θ to compute the features z_t (where the index t refers to the time index). In principle, any temporal action segmentation model can now be trained on top of these framewise features; we opt for the popular MS-TCN method (*Farha and Gall*, 2019) in our experiments in Chapter 6 and further evaluate the features using our UVAST method in Chapter 7.

3.2.3.1 Problem Formulation

Formally, the action segmentation task can be defined by a set of input videos, which are represented by sequences of framewise features $(z_1, \dots, z_T)$, and a target output sequence of action segments; the task is to train a model to map the set of framewise features to its corresponding sequence of action segments. The sequence of action segments can be represented as pairs of action labels and segment lengths, $((c_1, l_1), \dots, (c_k, l_k))$, or triplets of action labels, start and end times, $((c_1, s_1, e_1), \dots, (c_k, s_k, e_k))$, or equivalently, it can be given as a sequence of framewise action labels $(y_1, \dots, y_T)$. We can easily convert any of these representation forms into another. Current state-of-the-art methods predict framewise action labels, whereas we aim to directly predict the higher-level form of action label and segment lengths pairs in Chapter 7.

3.2.3.2 Evaluation Metrics

Next, we will introduce the metrics that we use throughout the thesis to evaluate the quality of the predicted segmentations. We denote the ground truth labels by $((\tilde{c}_1, \tilde{s}_1, \tilde{e}_1), \dots, (\tilde{c}_k, \tilde{s}_k, \tilde{e}_k))$ and $(\tilde{y}_1, \dots, \tilde{y}_T)$ for segment-level and frame-level label representation, respectively. Analogously, we denote the predicted labels by $((c_1, s_1, e_1), \dots, (c_m, s_m, e_m))$ and $(y_1, \dots, y_T)$.

Framewise Accuracy. The most commonly used evaluation metric is framewise accuracy (Acc), which simply computes the accuracy over all frames in the test set. Although, it gives a basic rating of the quality of the segmentation, it does not portray a thorough picture as it is insensitive to over-segmentation errors and results are dominated by longer action classes. Over-segmentation occurs when a longer segment is chopped up into several shorter ones due to only a few misclassified frames. Although, this scenario is clearly sub-optimal and can potentially be very problematic in applications, the impact of the few misclassified frames is relatively small compared with the many correctly classified frames in the segment. The same effect occurs for short vs. long action segments.

Segmental Edit Score. A more thorough evaluation metric which better reflects over-segmentation errors is the segmental Edit score, proposed by *Lea et al.* (2016b), which measures how well the model predicts the order of action segments. This metric evaluates the quality of the predicted transcript, *i.e.* $(c_1, \dots, c_m)$, and compares it with the ground truth transcript $(\tilde{c}_1, \dots, \tilde{c}_k)$ via the Levenshtein distance. Intuitively, the Levenshtein distance counts the minimum number of *edits*, *i.e.* insertion, deletion or substitution, required to transform one transcript into the other. To obtain the Edit score the Levenshtein distance is normalized by the maximum length of the two transcripts. Formally, we compute the Edit score between predicted and ground truth transcript as

$$\mathrm{Lev}((c_1,\dots,c_m),(\tilde{c}_1,\dots,\tilde{c}_k)) = \begin{cases} k & \text{if } m = 0, \\ m & \text{if } k = 0, \\ \mathrm{Lev}((c_2,\dots,c_m),(\tilde{c}_2,\dots,\tilde{c}_k)) & \text{if } c_1 = \tilde{c}_1, \\ 1 + \min \begin{pmatrix} \mathrm{Lev}((c_2,\dots,c_m),(\tilde{c}_1,\dots,\tilde{c}_k)), \\ \mathrm{Lev}((c_1,\dots,c_m),(\tilde{c}_2,\dots,\tilde{c}_k)), \\ \mathrm{Lev}((c_2,\dots,c_m),(\tilde{c}_2,\dots,\tilde{c}_k)) \end{pmatrix} & \text{otherwise} \end{cases}, \tag{3.19}$$

$$\mathrm{Edit}((c_1,\dots,c_m),(\tilde{c}_1,\dots,\tilde{c}_k)) = \left(1 - \frac{\mathrm{Lev}((c_1,\dots,c_m),(\tilde{c}_1,\dots,\tilde{c}_k)}{\max(m,k)}\right) \cdot 100. \tag{3.20}$$

Naturally, over-segmentation will be better reflected in this metric as each interruption of a segment will be present in the transcript as well – regardless of how short this interruption may be. Similarly, the Edit score is not impacted by the lengths of action segments. However, this measure can be insensitive to the alignment of predicted and ground truth segments since it does not take segment lengths into account. To get a better intuition, consider the extreme example: A predicted segmentation of $((c_1, 90), (c_2, 5), (c_3, 5))$ does not overlap well with the ground truth $((c_1, 30), (c_2, 40), (c_3, 30))$, but gives a perfect Edit score of 100.

Segmental F1 Scores. On the other hand, F1 scores, proposed in (*Lea et al.*, 2017), establish a more comprehensive measure as it is based on the high-level sequence of action segments but also takes the segment lengths into account. It penalizes over-segmentation and is insensitive to the lengths of action classes, while also reflecting alignment of action segments without being overly restrictive.

For each predicted segment $(\tilde{c}, \tilde{s}, \tilde{e})$ we determine whether it is a true or a false positive by taking a threshold ρ on its intersection over union (IoU) with the corresponding ground truth segment

(c, s, e), which is computed in the following way:

$$\text{IoU} = \frac{(c, s, e) \bigcap (\tilde{c}, \tilde{s}, \tilde{e})}{(c, s, e) \bigcup (\tilde{c}, \tilde{s}, \tilde{e})} = \frac{\min(\tilde{e}, e) - \max(\tilde{s}, s)}{\max(\tilde{e}, e) - \min(\tilde{s}, s)}. \tag{3.21}$$

If the IoU exceeds the given threshold ρ, we count it as a true positive; if there is more than one predicted segment that meets this criterion for a single ground truth segment, only one of them is counted as true positive and the others as false positives. Then, we can compute the precision and recall to obtain the F1 score given the total number of true positives (TP), false positives (FP) and false negatives (FN):

$$\text{precision} = \frac{TP}{TP + FP}, \tag{3.22}$$

$$\text{recall} = \frac{TP}{TP + FN}, \tag{3.23}$$

$$\text{F1} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{3.24}$$

In Chapters 6 and 7, we report the F1 scores at different overlapping thresholds $10\%, 25\%, 50\%$, denoted by F1@10, F1@25, F1@50, respectively. By taking thresholds we make sure not to penalize small temporal shifts between the predicted and ground truth segments; the boundaries between actions are naturally blurry and the perceived ground truth boundary may vary from person to person, therefore, small shifts do not indicate a false prediction.

3.3 Neural Network Architectures for Video Models

Modern video understanding models are based on deep neural networks. In the following, we will recap the neural network architectures that we use throughout the thesis. Our backbone networks in Chapters 4, 5 and 6 are based on different variants of ResNet (*He et al.*, 2016), which we revisit in Section 3.3.1. Additionally, our model in Chapter 4 uses a gated recurrent model, which we review in Section 3.3.2, on top of the ResNet backbone to aggregate features. Our model in Chapter 7 on the other hand, is based on a Transformer model (*Vaswani et al.*, 2017), see Section 3.3.3.

3.3.1 Deep Residual Networks

Arguably, one of the most famous deep learning architectures is the deep residual network (ResNet) (*He et al.*, 2016), which achieved state-of-the-art performance on several image classification and detection tasks and has become one of the go-to models for many computer vision tasks. The core idea of ResNets is the residual layer with an identity shortcut connection that adds the output of the previous layer to the output of the current layer. As a result, the layers do not need to fit an underlying mapping directly but only the residual mapping, which is easier to optimize and enables the training of very deep networks. Due to their immense success on image related tasks, ResNets have been adapted to the video domain as well (*Hara et al.*, 2018). We will first review the traditional ResNet architecture designed for images, which we use in the experiment section of Chapter 5, and then discuss the adaptations to video input.

Layer Name	ResNet-18	ResNet-50	(2D)3D-ResNet18
conv1	$7 \times 7, 64$, stride 2		$\{1, 7\} \times 7 \times 7, 64$, stride $(1, 2, 2)$
conv2_x	3×3 max pool, stride 2 $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\{1, 3\} \times 3 \times 3$ max pool, stride $(\{1, 2\}, 2, 2)$ $\begin{bmatrix} \{1, 3\} \times 3 \times 3, 64 \\ \{1, 3\} \times 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} \{1, 3\} \times 3 \times 3, 128 \\ \{1, 3\} \times 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix} \times 2$
fc	global avg pool, n-dim. fully connected, softmax		

Table 3.1: **ResNet Architectures.** Each convi_x block denotes a residual block, consisting of several convolutional layers; $k \times k, C$ specify their kernel size and output channels, respectively.

A single residual block is composed of a function approximating the residual function and the identity:

$$y = f_{\theta_i}(x) + x, \tag{3.25}$$

where x and y are the input and output of the block, respectively. In some cases, the output dimension of $f_{\theta_i}(x)$ can be different from x and we need to modify Eq. (3.25) to

$$y = f_{\theta_i}(x) + W_i x, \tag{3.26}$$

using a linear projection with weights W_i. The function f_{θ_i} typically consists of several convolutional layers, followed by BatchNorm (*Ioffe and Szegedy*, 2015) and ReLU activation. The detailed architectures of ResNet-18 and ResNet-50 can be found in Table 3.1. The first conv1 layer consists of a single 7×7 convolution with 64 output channels applied with a stride of 2. Each convi_x block in Table 3.1 denotes a residual block, consisting of several convolutional layers, where $k \times k$ and C specify the kernel size and output channels of the convolutional layer, respectively. After the last residual block, the feature map is global average pooled into a single feature vector and a linear classification head is applied for classification.

Hara et al. (2018) adapt this architecture to the video domain by modifying the 2D convolutional layers to 3D ones, *i.e.* the $k \times k$ filters are altered to $k \times k \times k$. Furthermore, the temporal stride in conv1 is reduced to 1, while the spatial stride remains 2. While *Hara et al.* (2018) propose to replace all 2D convolutions with 3D convolutions, intermediate adaptations are possible. The 2D3D-ResNet version used in (*Han et al.*, 2019) that we adopt for our experiments in Chapter 4, keeps the 2D convolutional layers in conv1, conv2_x and conv3_x (*i.e.* with kernel size $1 \times k \times k$), and only replaces those in conv4_x and conv5_x with 3D filters ($k \times k \times k$).

3.3.2 Gated Recurrent Units

Recurrent neural networks (RNNS) are especially useful when dealing with variable length data, such as language or videos. They process one element of the sequence at a time and keep a recurrent hidden state that is dependent on the previous elements in the sequence. Vanilla RNNs are difficult to train to capture long-term dependencies over the sequence; two popular models that overcome this problem are long short-term memory (LSTM) networks (*Hochreiter and Schmidhuber*, 1997) and gated recurrent units (GRUs) (*Cho et al.*, 2014). In Chapter 4, we use a GRU to aggregate features of several consecutive video clips into a single feature vector.

GRUs use a gating mechanism that controls the information flow: an update gate that controls how the hidden state is updated and a reset gate that modulates the current hidden state. More specifically, given an input sequence of clip-level features $(z_1, \dots, z_n)$ and an initial hidden state h_0, we compute the update gate u_t and reset gate r_t

$$u_t = \sigma(W_u[z_t, h_{t-1}]), \tag{3.27}$$

$$r_t = \sigma(W_r[z_t, h_{t-1}]), \tag{3.28}$$

where W_u, W_r define linear mappings and σ is computed element-wise. Then, a candidate activation $\tilde{h}_t$ is computed

$$\tilde{h}_t = \tanh(W[z_t, r_t \odot h_{t-1}]), \tag{3.29}$$

where $\odot$ is element-wise multiplication. The new hidden state h_t is a linear interpolation between the previous h_{t-1} and the candidate activation $\tilde{h}_t$:

$$h_t = (1 - u_t)h_{t-1} + u_t\tilde{h}_t. \tag{3.30}$$

The final hidden state h_n provides an aggregated global representation of the full input sequence $(z_1, \dots, z_n)$.

3.3.3 Transformer

The Transformer architecture (*Vaswani et al.*, 2017) has become the de-facto architecture for many natural language processing tasks, and has recently achieved impressive results on vision related tasks as well (*Dosovitskiy et al.*, 2021; *Liu et al.*, 2021a). The vanilla Transformer architecture takes as input a sequence of (word) tokens, forwards them through several self-attention layers in the encoder module, and then translates them to the output sequence using an auto-regressive decoder with cross-attention layers. The self- and cross-attention layers allow the Transformer to model dependencies between each element in the sequence and all other elements. In the following, we first introduce the basic building blocks of Transformers before diving into the details of our architecture used in Chapter 7.

Attention. The attention mechanism is the most important component of Transformers – both encoder and decoder layers are built on it. The attention mechanism takes a set of n query vectors $Q \in \mathbb{R}^{n \times d_k}$, each of dimension d_k, and m key-value pairs $K \in \mathbb{R}^{m \times d_k}, V \in \mathbb{R}^{m \times d_v}$. We compute the attention between Q, K, V as

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{3.31}$$

Intuitively, the dot-product inside the softmax evaluates the similarities between all queries and keys – this allows all elements in the sequence to interact with all other elements – and computes a weighted average of the values based on those similarities.

In most cases, several such attention mechanisms are applied with multiple heads, referred to as multi-head attention. Each head_i linearly projects the d_{model}-dimensional queries, keys and values to dimension d_k, d_k, d_v, using projection matrices $W_i^Q, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$ and $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$. The outputs of all heads are concatenated and projected back to the original dimension using a projection matrix $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$:

$$\text{MHA}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O, \tag{3.32}$$

$$\text{head}_i = \text{Attn}(QW_i^Q, KW_i^K, VW_i^V). \tag{3.33}$$

We use two different variants of attention in the encoder and decoder model, which mainly differ based on which input is used as queries, keys and values. Namely, the encoder model employs so-called self-attention layers, where Q, K, V are linear projections of the output of the previous layer, *i.e.* they are obtained from the same sequence. While the decoder employs self-attention as well, it additionally contains cross-attention layers to enable information flow between the encoded input sequence of the encoder model and the output sequence of the decoder model: Here, the queries are obtained from the previous decoder layer and the keys and values come from the encoder output.

Encoder. Suppose $\tilde{x} \in \mathbb{R}^{n \times d_{\text{model}}}$ is the input to the current encoder layer. We first compute the multi-head attention of the projected queries, keys and values using $W^Q, W^K, W^V \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$, followed by LayerNorm (*Ba et al.*, 2016) and a residual connection:

$$\tilde{x}_{\text{mha}} = \text{MHA}(\tilde{x}W^Q, \tilde{x}W^K, \tilde{x}W^V), \tag{3.34}$$

$$\hat{x} = \text{LayerNorm}(\tilde{x} + \tilde{x}_{\text{mha}}). \tag{3.35}$$

We then apply a two-layer fully connected feed forward network (FFN) with `ReLU` activation between the two layers, again followed by LayerNorm and a residual connection to obtain the output x_{out} of the layer:

$$x_{\text{out}} = \text{LayerNorm}(\hat{x} + \text{FFN}(\hat{x})). \tag{3.36}$$

The complete encoder model consists of N such layers E_n that are applied consecutively. Given the embedded input sequence $x \in \mathbb{R}^{n \times d_{\text{model}}}$, it first supplies x with a sinusoidal positional encoding pe of the same dimension as x and then passes it through all N layers to obtain encoded features e:

$$e = (E_N \circ \cdots \circ E_1)(x + pe). \tag{3.37}$$

With the help of the self-attention mechanism the encoder can model long-range dependencies in e between all elements of the input sequence x. While the traditional Transformer model used in the natural language processing domain takes as input a sequence of word tokens, our model in Chapter 7 processes a sequence of frame-wise features $z \in \mathbb{R}^{n \times d}$. We embed this input sequence using a linear layer $L : \mathbb{R}^d \to \mathbb{R}^{d_{\text{model}}}$, which we apply to each feature vector in the sequence independently.

Decoder. The task of the decoder is to translate the input sequence to a desired output sequence. It receives two inputs: The output $e \in \mathbb{R}^{n \times d_{\text{model}}}$ of the encoder model and the current embedded output sequence $y \in \mathbb{R}^{k \times d_{\text{model}}}$ up to element k (the first element if a start-of-sequence token). A single layer takes the output of the previous layer $\tilde{y}$ and e, and similar to the encoder first computes multi-head self-attention and residual connection on $\tilde{y}$:

$$\tilde{y}_{\text{mha}} = \text{MHA}(\tilde{y}W^Q, \tilde{y}W^K, \tilde{y}W^V), \tag{3.38}$$

$$\hat{y} = \text{LayerNorm}(\tilde{y} + \tilde{y}_{\text{mha}}). \tag{3.39}$$

Since we apply the decoder auto-regressively during inference, we want to prevent acausal information flow, which is typically achieved by masking out all values in the attention that form acausal connections. Next, a cross-attention mechanism between $\hat{y}$ and e is employed. Here, the queries come from the decoder features $\hat{y}$, while keys and values come from the encoder features e:

$$\hat{y}_{\text{mha}} = \text{MHA}(\hat{y}W^Q, eW^K, eW^V), \tag{3.40}$$

$$\hat{y}_{\text{CA}} = \text{LayerNorm}(\hat{y} + \hat{y}_{\text{mha}}), \tag{3.41}$$

followed by the feed forward network and residual connection to obtain the output y_{out} of the layer:

$$y_{\text{out}} = \text{LayerNorm}(\hat{y}_{\text{CA}} + \text{FFN}(\hat{y}_{\text{CA}})). \tag{3.42}$$

The decoder supplies the sequence y with a sinusoidal positional encoding pe first, and then passes it together with the encoded features e through N decoder layers D_n.

$$z = (D_N \circ \cdots \circ D_1)(y + pe, e). \tag{3.43}$$

Finally, a linear layer and softmax are applied on z to obtain the output probabilities.

For our transcript decoder and alignment decoder in Chapter 7, we use 2 and 1 layers, respectively.

Hierarchical Encoder. The above models constitute the vanilla Transformer model of (*Vaswani et al.*, 2017). For our experiments in Chapter 7, we further use an encoder model inspired by (*Yi et al.*, 2021), which modifies the vanilla encoder in several ways. The modified model is illustrated in Figure 3.2. First, we insert a dilated temporal convolution with dilation rate 2^i in each layer i, followed by `GeLU` activation. Then, we apply InstanceNorm (*Ulyanov et al.*, 2017) instead of LayerNorm *before* the single-head self-attention instead of after. The feed forward network is replaced by a 1×1 convolution, followed by another dilated convolution and `GeLU` activation. LayerNorm is dropped in this step.

The most significant modification is within the self-attention layer: In contrast to natural language processing, our input sequences are much longer, consisting of thousands of frames, which makes it very difficult for the model to focus on meaningful locations in the sequence. Therefore, the self-attention is restricted to a local window around the query. We use windows of size $2^{i+1} + 1$ in the i-th layer. More specifically, we adjust the attention in Eq. (3.31) in the following way. Recall, that the self-attention mechanism is computed on a set of n query, key and value vectors $Q = \tilde{x}W^Q \in \mathbb{R}^{n \times d_{\text{model}}}$, $K = \tilde{x}W^K \in \mathbb{R}^{n \times d_{\text{model}}}$, $V = \tilde{x}W^V \in \mathbb{R}^{n \times d_{\text{model}}}$. To compute the attention of the j-th

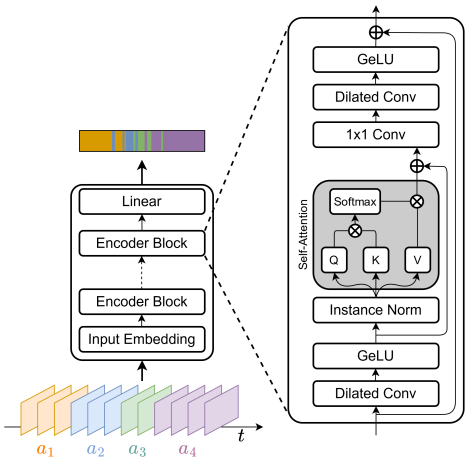


Figure 3.2: **Modified Transformer Encoder Model.** In Chapter 7, we use a modified version of the encoder model proposed in (*Yi et al.*, 2021). Specifically, we modify the encoder block of the vanilla Transformer model in several ways: First, we insert a dilated convolution, followed by a `GeLU` activation before computing the self-attention. Instead of LayerNorm we use InstanceNorm (*Ulyanov et al.*, 2017) *before* the single-head self-attention rather than after. We further replace the FFN by a 1×1 convolution, followed by another dilated convolution and `GeLU` activation. Furthermore, we restrict the self-attention mechanism to a local window around the query.

query Q_j we consider the keys and values in a local window around j, *i.e.* $(K_{j-2^i}, \ldots, K_{j+2^i})$ and $(V_{j-2^i}, \ldots, V_{j+2^i})$:

$$\text{Attn}(Q, K, V)_j = \text{softmax}\left(\frac{Q_j (K_{j-2^i}, \ldots, K_{j+2^i})^T}{\sqrt{d_{\text{model}}}}\right)(V_{j-2^i}, \ldots, V_{j+2^i}). \tag{3.44}$$

Compared to the encoder model in (*Yi et al.*, 2021) this version uses `GeLU` instead of `ReLU` activations and uses an additional dilated convolution followed by `GeLU` at the end of the encoder block. We use 10 layers in total in this modified encoder.

CHAPTER 4

Self-Supervised Video Representation Learning by Bidirectional Feature Prediction

In this chapter, we introduce a method for self-supervised video representation learning based on bidirectional feature prediction. While future prediction has been widely explored as a pretext task, the unobserved past has been largely overlooked – although it may provide an equally useful source of supervision as future prediction. To fill this gap, we propose a method that takes advantage of both future and past frames to derive a challenging task, which encourages the model to focus on the temporal structure of videos. Given a set of present frames, the model should not only be able to reason what has happened prior to the current events and what is going to happen in the immediate future, but also be able to distinguish between past and future events. We pose this task in a contrastive learning setting, where we ask the model to jointly predict past and future features and distinguish between a reversed order of future and past via temporal hard negatives. We demonstrate the effectiveness of our method empirically via several downstream tasks.

Individual Contribution

The following chapter is based on the publication (*Behrmann et al.*, 2021b):

Unsupervised Video Representation Learning by Bidirectional Feature Prediction
Nadine Behrmann, Juergen Gall, and Mehdi Noroozi
IEEE Winter Conference on Applications of Computer Vision (WACV), 2021.

This publication was done in very close collaboration between Mehdi Noroozi and Nadine Behrmann. Juergen Gall provided scientific guidance and supported this work with very valuable feedback and suggestions. The initial idea to exploit unobserved past frames for self-supervised video representation learning was proposed by Mehdi Noroozi and was initially targeted mostly towards distinguishing between past and future given present. The method was then further refined in joint discussions between Mehdi Noroozi and Nadine Behrmann. The empirical evaluation was implemented and carried out by Nadine Behrmann.

Contents

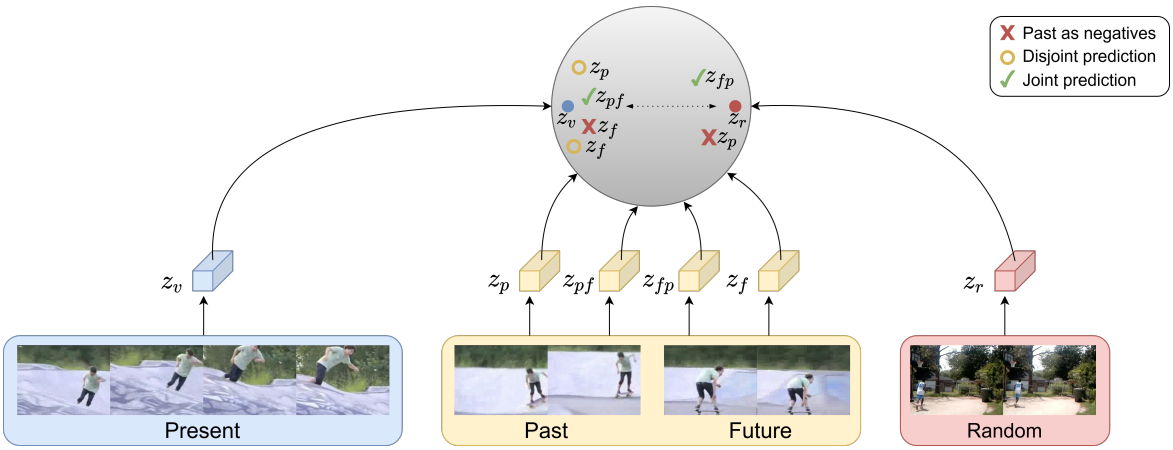


Figure 4.1: **Effect of Past and Future Prediction on the Feature Space.** The proximity of the features depicted in the gray feature space indicates their (dis-)similarity. For an ideal embedding space we desire that I) past, present and future are similar and II) past and future are distinguishable given present. We consider three approaches to construct positives and negatives for the present feature z_v: **Disjoint prediction** (yellow circles), which only satisfies (I), requires past z_p and future z_f to be distinguishable from random z_r. **Past as negatives** (red crosses) only fulfills (II). Here, distinguishing past and future comes at the cost of dissimilarity of the past feature to present and future, implicitly violating (I). Only our **Joint prediction** (green ticks) satisfies both (I) and (II). Here, the combination of past and future in the right order z_{pf} should be distinguishable from random as well as the wrong order z_{fp}, (promoting similarity of past, present and future, while distinguishing the right and wrong order of future and past.)

4.1 Introduction

Videos provide a rich source of information for visual understanding. They generously reveal to our machines how objects interact with each other and the environment in the real world. To utilize this information, the task of representing high-level abstractions from videos is essential to address a large and sophisticated set of downstream tasks tied to videos, *e.g.* temporal action segmentation (*Richard et al.*, 2017).

Self-supervised learning has recently established a promising direction for this purpose. A well known motivation for self-supervised learning from a practical point of view is to alleviate the cost and error of the labeling process. Moreover, learning generalizable and optimal representations can not be taken for granted in a supervised setting – especially when representing a complex source of information like videos. For instance, it has been shown that action labels are predictable from a single frame to an acceptable extent (*Wang et al.*, 2019b; *Fouhey et al.*, 2018), providing a relatively weak source of supervision signal for representation learning as the network is not forced to explore temporal information of videos. In fact, the commonly used datasets are inherently biased towards static features (*Li et al.*, 2018), allowing the model to largely ignore the temporal structure of videos, which, however, can be detrimental for other downstream tasks that rely on temporal information.

In contrast to the image domain, where the gap between self-supervised and supervised representation learning has been shrunk remarkably (*Chen et al.*, 2020b; *Asano et al.*, 2020; *Grill et al.*, 2020), self-supervised video representation learning still lags behind supervised learning even regarding relatively simple tasks such as action recognition on short trimmed videos. Nevertheless, the temporal structure and multi-modal nature of videos provide even more opportunities to construct pretext tasks compared with images. While recent work on self-supervised multi-modal video representation learning has shown to be very effective (*Piergiovanni et al.*, 2020), we are interested in RGB-only self-supervised video representation learning. Besides the scientific value of pure vision based models, a practical motivation involves applications where the audio signal is not accessible, *e.g.* autonomous driving.

We introduce a novel self-supervised task based on predicting the features of unobserved parts of the data, *i.e.* past and future frames. Previous approaches of learning from prediction have been limited to future observations: They aim to train a model that takes a video clip as input and predicts some form of the contents of future frames (*Mathieu et al.*, 2016). Our main idea involves incorporating *unobserved past* – an element that has been largely ignored by previous works – and future jointly. Given a clip of a video as a present sequence, the question of what will happen in the future frames is comparable to asking what has happened in the past in terms of abstract factors of variation which the network needs to encode. We propose to exploit both past and future to design an even more challenging pretext task for enriched representations.

Nonetheless, utilizing both signals is not trivial: Simply predicting past and future independently does not stimulate the network to enhance the representations. We attribute this to the high similarity between the tasks – the features which enable the prediction of the future are also sufficient to predict the past. Another explanation can be found when viewing this task from the contrastive learning perspective, specifically considering the role of the negatives: The feature prediction task involves maximizing mutual information approximated mostly via a contrastive loss that compares joint and product of the marginals of present and future/past distributions. The quality of learned representations is highly dependent on the set of negatives (*Tschannen et al.*, 2020b). While random sequences form the basis for negatives, they are easily distinguishable from the matching pair via shortcuts such as low-level statistics, edges, etc. To prevent these shortcuts, the key ingredient for a comprehensively challenging pretext task is to construct additional negatives that are hard to distinguish – suitably referred to as *hard negatives* – encouraging the network to explore the structure of the data more intensively. Disjoint prediction of past and future does not introduce a new set of negatives compared to individual prediction of future or past, see Figure 4.1.

Instead, we propose to predict future and past jointly in a **B**idirectional **F**eature **P**rediction task

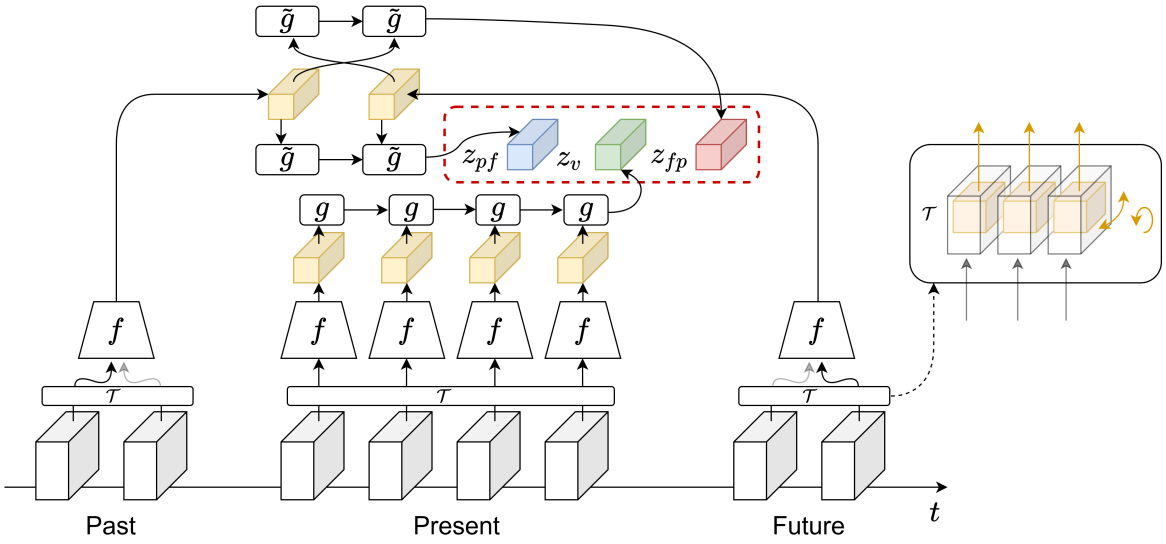


Figure 4.2: **Bidirectional Feature Prediction.** First we apply spatial transformations $\mathcal{T}$ to past, present and future independently, these can include random crop, horizontal flip and rotation. Then a 3D CNN f is applied to the non-overlapping clips of frames. To obtain a video representation z_v an aggregation function g is applied to the sequence of features. We randomly sample a past and a future clip, which is indicated by the black and gray arrows, and apply an aggregation function $\tilde{g}$ to acquire a merged representation z_{pf} of past and future. A temporal hard negative z_{fp} is obtained by applying $\tilde{g}$ to the swapped features of future and past. To maximize the agreement of z_v and z_{pf} we employ the InfoNCE loss where we incorporate z_{fp} as temporal hard negative in the denominator.

(BFP). Our method establishes a connection between present frames with a pair of future and past frames. This allows us to incorporate the wrong order of future and past as hard negatives: Given present frames, the network should distinguish the corresponding pair of future and past not only from randomly taken pairs but also the swapped future and past of the same video, forcing the model to focus on the temporal structure of the present frames. Our experiments show that joint prediction of future and past outperforms disjoint prediction on several transfer learning benchmarks.

Our contributions include: 1) We propose a novel bidirectional feature prediction pretext task for video representation learning. 2) We extensively evaluate our proposed method on several transfer learning benchmarks showing its superiority to its future prediction counterpart.

4.2 Method

The main motivation for our **B**idirectional **F**eature **P**rediction method (BFP) is to incorporate unobserved past frames in addition to future frames into a challenging pretext task to encourage the model to capture the temporal structure of videos. A straight-forward prediction of past and future is problematic as the supervisory signals arising from true data distributions suffer from ambiguity and require the network to devote substantial capacity to model a data distribution, which is not necessarily the optimal solution for representation learning. While this issue can be resolved by predicting the features instead of raw pixels, unobserved past and future are inherently uncertain – for

example a person standing at a cross walk may have previously exited the book store next door or simply walked down the sidewalk and, similarly, may decide to cross the street or continue down the sidewalk – making the prediction quantification challenging.

To handle this stochasticity without imposing a parametric form, we employ a variant of Noise Contrastive Estimation, *InfoNCE* (*van den Oord et al.*, 2018), in which the network is asked to distinguish the positive pair – composed of the predicted features and the actual target features – from a set of negatives. This allows the model to capture the multi-modal distribution of the target features without imposing it explicitly. More precisely, given a video clip we divide the video frames into three partitions, $X = (P, V, F)$, each element refers to past, present, and future, respectively. We then construct the positive and negative pairs to train InfoNCE by exploiting the joint representations of (P, F) and leverage a temporal hard negative (based on the reversed order of future and past) to encourage temporally structured representations.

4.2.1 Past and Future Feature Prediction

First, we explore how the two complementary supervision signals can be incorporated into a comprehensively challenging pretext task. This seemingly easy objective proves to be non-trivial and deserves an in-depth discussion. A naive approach to combine past and future prediction will be in a disjoint fashion: The task of past and future prediction is treated independently in a unidirectional manner by simply adding the respective losses. This is equivalent to encouraging distinguishable features of both future and past from random given present. Unfortunately, this straightforward approach does not achieve decent results, see Table 4.1. We conjecture that representations, which are able to predict the future, are sufficient to predict the past and thus the added losses do not stimulate the network to enhance the representations.

Instead, to encourage the model to explore the temporal structure further, we construct a pretext task that involves distinguishing between future and past, given present. We did not achieve reasonable results by explicitly classifying future, past, and random pairs. Alternatively, we use the InfoNCE loss and implicitly exploit this signal. An apparent choice is to include past features as negative pairs, while future features form the positive pairs – effectively instructing the model to distinguish between future and past. However, this is not an appropriate approach. Essentially, past features should encode similar high-level global abstractions of the video (such as underlying action, objects and people involved in the video) as those of the future. Using past as negatives entails dissimilarity between representations of the past and present/future, see Figure 4.1, resulting in a degenerate solution that removes such high-level abstractions shared across past, present, and future from the representations. Our experiment termed *Past as negatives* in Table 4.1 confirms the poor quality of the resulting representations of this approach. A desired solution favors distinguishable features of future and past *given present*: With access to information of current events a distinction between past and future events is achievable. To this end, we propose joint prediction of future and past. Our model takes a pair of past/future as input, which allows temporal hard negatives of the wrong order of future/past. Distinguishing between past/future and future/past requires the network to encode temporal structure shared across the video such that matching temporal orders can be detected. Our approach encourages the features of future and past to be distinguishable from random as well as each other given present.

4.2.2 Training Objective

For a given video, we build a set $\mathcal{P}$ consisting of positive future and past pairs. For each positive pair $(P, F) \in \mathcal{P}$ we consider a set $\mathcal{N}_{(P,F)}$ containing all negative samples, including the reverse order of future and past (F, P). We denote the features of the present clip V by z_v, and those of past/future (P, F) and future/past (F, P) by z_{pf} and z_{fp}, respectively. As suggested in (*Chen et al.*, 2020b), we use a small MLP head h to map the representations to the space in which the contrastive loss is applied. We use the cosine similarity $\mathrm{sim}(z_1, z_2) = \frac{1}{\tau} \frac{h(z_1)^T h(z_2)}{\|h(z_1)\| \|h(z_2)\|}$ (see Eq. (3.5)) with a temperature parameter τ to compare present features with those obtained from future and past,

$$\mathcal{L} = \sum_{(P,F)\in\mathcal{P}} -\log \left(\frac{\exp(\mathrm{sim}(z_v, z_{pf}))}{\exp(\mathrm{sim}(z_v, z_{pf})) + \sum_{N\in\mathcal{N}_{(P,F)}} \exp(\mathrm{sim}(z_v, z_n))} \right). \tag{4.1}$$

We discuss the process of constructing positive and negative pairs in the following.

Positives. $\mathcal{P}$ denotes a set of positive past/future clips. We obtain multiple positive pairs per video by selecting a random clip from past and future clips, respectively. If there are m past and future clips per video, we build m^2 positive pairs per video, see Figure 4.2.

Easy Negatives. We obtain easy negatives by sampling all possible combinations of past/future and future/past clips from other videos in a batch. If there are m past and future clips per video, a batch size of n provides us with $2m^2(n-1)$ easy negatives. The factor 2 emerges due to past and future swapping.

Temporal Hard Negatives. Temporal hard negatives are obtained via swapping the order of past and future. Consequently, we obtain the same number of temporal hard negatives per video as the number of positives. Each set $\mathcal{N}_{(P,F)}$ contains all temporal negatives.

Independent Augmentation vs. Spatial Negatives. *Han et al.* (2019) use spatial negatives in their method by decomposing the $4 \times 4 \times 256$ convolutional feature map along the depth direction into 16 feature vectors of size 256, and treat all of those that do not match the corresponding spatial location as negatives. We propose to simply apply independent (spatial) augmentations to P, V, F, and consider a single 256-dimensional feature vector via global average pooling. Our experiments show that simple independent augmentations achieve higher performance than complex spatial negatives. This is inline with recent observations made by *Misra and van der Maaten* (2020). Spatial negatives aim to learn features *variant* to the spatial location; independent augmentations aim to learn features *invariant* to the spatial location. Moreover, the major drawback of spatial negatives is the following: They encourage the feature vectors to represent local descriptors. The feature vectors should be distinguishable across spatial locations of the feature map since they are injected as negatives in the loss function. This assumption might be useful in the early layers where the receptive fields of neurons are small. However, in the later layers where the receptive field grows, a global feature is favorable – otherwise high-level features that are shared across the patches run the risk of being removed from the representation. Our experiments in Table 4.3 verify that indeed the features learned via spatial negatives transfer poorly to the downstream task in the later layers.

4.2.3 Connection to Mutual Information Maximization

The InfoNCE loss has been shown to be a lower bound of mutual information (*Poole et al.*, 2019). From this point of view, our loss in Eq. (4.1) can be interpreted as mutual information maximization of the features extracted by f:

$$\max_{f} \quad I(f(V), f(P, F)). \tag{4.2}$$

A correct lower bound is obtained when the negative pairs are built via sampling from the product of the marginals. Our temporal hard negatives are not sampled independently from the present clips, imposing an incorrect approximation of the mutual information. However, this is not counterproductive. As it has been recently shown (*Tschannen et al.*, 2020b), the effectiveness of the InfoNCE loss for learning representation is not necessarily tied to the accuracy of mutual information estimation. Structured hard negatives which are not sampled from the product of the marginals contribute to the quality of the learned representations more than an accurate estimation of mutual information. We could not achieve decent results without temporal hard negatives, see Table 4.6, confirming the same argument.

4.3 Experiments

Following the common practice in self-supervised video representation learning, we evaluate our approach on the downstream task of action recognition. During the self-supervised pretraining stage we discard the labels of the dataset and train a model using Eq. (4.1). Note that although we drop the class labels, a supervision bias remains in the dataset, *e.g.* the videos in Kinetics-400 are temporally trimmed and carefully selected. However, this is a widely adopted approach and we follow previous works in order to do fair comparisons. To evaluate the learned representation, we transfer the pretrained network, including the 2D3D-Resnet18 backbone f and aggregation function g. We add a linear layer on top of the resulting feature vector and evaluate our network for action recognition on labeled data. We follow two paradigms: 1) We *finetune* the entire network with randomly initialized linear classification layer for the task of action recognition, using a reduced learning rate for pretrained layers. 2) Additionally, we evaluate the quality of the representations layer-wise by *freezing* the pretrained network up to some layer, and training the remaining from scratch. This evaluation was primarily proposed by (*Noroozi and Favaro*, 2016) in the image domain. Our experiment on Kinetics-400 allows a more comprehensive evaluation of the quality of learned representations as it gives a better insight into the learned representations. It is well known that the early layers of CNNs extract general and local features whereas later layers are more specific to the training task and dataset (*Yosinski et al.*, 2014). A better performance on the downstream task of the later layers indicates higher correlation of pretext and downstream task.

4.3.1 Implementation Details

Network Architecture. For fair comparisons, we use a 2D3D version of ResNet18 as in (*Han et al.*, 2019) that consists of 3D convolutions only in the last two layers. The video sequence is divided into non-overlapping blocks of frames. We extract convolutional features from the blocks via a shared 2D3D-ResNet18, f. To create the final features, we aggregate the extracted convolutional features via

one-layer Convolutional Gated Recurrent Units (ConvGRU) with kernel size 1. We found that using a separate aggregation function for past/future blocks achieves slightly better results, see Table 4.5. Consequently, we construct the final feature of present blocks as $z_v = (g \circ f)(V)$, and past/future blocks as $z_{pf} = (\tilde{g} \circ f)(P, F)$, where g, $\tilde{g}$ denote aggregation functions of present and past/future blocks respectively. As $\tilde{g}$ is non-symmetric, we can apply it to the reverse order of future/past to obtain a temporal hard negative $z_{fp} = (\tilde{g} \circ f)(F, P)$, which will be discussed below. Figure 4.2 shows all components of our method.

Implementation Details. We use similar self-supervised learning and evaluation settings to (*Han et al.*, 2019), our most important baseline. We extract 8 blocks of 5 frames from a video sequence and split them in the following way. The 4 middle blocks are used as present video sequence, while we sample single past and future blocks at different temporal distances to the present from the remaining blocks. In this case, we have 4 positive pairs per video and 4 corresponding negative pairs. During self-supervised learning we apply random crop and horizontal flip, as well as frame-wise color jittering and random downsampling of the frame rate, where we sample video frames with a random stride of at most 3. For finetuning we keep the random crop and horizontal flip, but apply consistent color jittering at the video-level and sample frames with a constant stride of 3. In the case of spatial negatives, the spatial transformations crop and horizontal flip are applied consistently to the entire video. As discussed above, we propose to augment past, present, and future video sequences independently, resulting in more robust representations. We conduct experiments with both settings. Note that spatial negatives can not be combined with independent spatial transformations since they require spatial correspondence. During self-supervised training, we map the features to the contrastive space using a small MLP head, consisting of 256 hidden units and a ReLU activation function. After self-supervised training, the MLP is discarded and only the backbone and aggregation function are transferred to the downstream task. For the downstream task, we use the same input structure of video blocks as during self-supervised training. For both self-supervised training and finetuning, we use a batchsize of 64 and the Adam optimizer (*Kingma and Ba*, 2015) with a learning rate of 10^{-3} and weight decay 10^{-5}. We reduce the learning rate by a factor of 10 when the validation loss plateaus. For the layer-wise evaluations we use SGD with an initial learning rate of 0.1, a weight decay of 10^{-3}, a momentum of 0.9 and a learning rate scheduler. For self-supervised training, we train the network for 100 and 470 epochs on Kinetics-400 and UCF101, respectively. For finetuning, we follow (*Han et al.*, 2019) and fix the number of epochs to 300 and reduce the learning rate to 10^{-4} for the pretrained weights. During inference, we divide videos from the validation set into blocks of 5 frames and construct half-length overlapping sequences of blocks that are fed to the model. We remove augmentations and only use center crop. We average the softmax probabilities to obtain the final classification scores.

4.3.2 Finetuning on UCF101 and HMDB51

A most widely used framework of self-supervised learning involves obtaining an initialization from a large scale unlabeled dataset, and finetuning on a small annotated target dataset. We consider UCF101 (*Soomro et al.*, 2012) and HMDB51 (*Kuehne et al.*, 2011) as the standard benchmarks in this domain. We report numbers on split 1 for both datasets.

Method	top1 Accuracy on UCF101
Future prediction	61.3
Past prediction	60.1
Disjoint prediction	60.1
Past as negatives	57.6
Ours	**63.6**

Table 4.1: **Past and Future Prediction.** Experiments were conducted on UCF101 (both for pretraining and finetuning), using spatial negatives. Future prediction refers to the DPC method trained with our implementation details, while Past prediction denotes the DPC method applied to predicting past features. Disjoint prediction constitutes the added losses of past and future prediction and Past as negatives refers to InfoNCE trained with future features as positives and past features as negatives.

Pretraining on UCF101. The main motivation of self-supervised learning is to take advantage of large scale unlabeled datasets. Therefore, pretraining on a small dataset such as UCF101 does not establish a plausible framework. Following the community, we aim to quickly validate the design choices of our bidirectional feature prediction task in the following experiments. To that end, we pretrain different self-supervised tasks on the UCF101 dataset and evaluate the learned representations via finetuning on the same dataset for the task of action recognition. We show the results in Table 4.1. All methods are trained using the same pretraining hyperparameters, including network architecture, number of pretraining epochs, and finetuning strategy in order to allow conclusive and fair comparisons. For the same reason, we employ spatial negatives for all methods in this specific experimental setup. First, we compare our method to past and future prediction. The future prediction corresponds to the DPC method using our implementation details, which improves slightly over the original implementation (*Han et al.*, 2019). The past prediction is the same task but the unobserved past frames are used instead of future frames. While both future prediction and past prediction learn representations that prove to be useful for action recognition, simply adding the two losses (*Disjoint prediction*) does not improve the quality of the representations, suggesting that both tasks can be solved with a similar set of features. Another naive approach of incorporating both supervisory signals is to use past features as negatives in the InfoNCE loss, while only the future features are treated as positives, which implicitly requires the network to distinguish between past and future. However, this approach leads to weaker representations by removing high-level abstractions shared across past, present and future, shown as *Past as negatives* in Table 4.1. This demonstrates that unseen past frames provide a valuable supervision signal, but incorporating both past and future prediction in a comprehensively challenging task is non-trivial. In contrast, our method based on bidirectional feature prediction improves the performance on the downstream task of action recognition, validating our design choices. While we outperform unidirectional prediction, our method is slightly inferior to VCOP (*Xu et al.*, 2019), see Table 4.2. However, note that VCOP employs the R3D architecture that includes 3D convolutions in all layers whereas our 2D3D-Resnet18 uses 3D convolutions only in the last two layers. Their network consumes more memory and computations compared to ours.

Self-Supervised Methods			top1 Accuracy	
Method	Architecture	Pretraining Dataset	UCF101	HMDB51
Shuffle&Learn (*Misra et al.*, 2016)	CaffeNet	UCF101/HMDB51	50.2	18.1
OPN (*Lee et al.*, 2017)	VGG	UCF101/HMDB51	59.8	23.8
O3N (*Fernando et al.*, 2017)	AlexNet	UCF101	60.3	32.5
VCOP (*Xu et al.*, 2019)	R3D	UCF101	64.9	29.5
3DRot (*Jing and Tian*, 2018)	3D-R18	Kinetics-600	62.9	33.7
3D-ST-Puzzle (*Kim et al.*, 2019)	3D-R18	Kinetics-400	65.8	33.7
VIE (*Zhuang et al.*, 2020)	3D-R18	Kinetics-400	72.3	44.8
LA-IDT† (*Tokmakov et al.*, 2020)	3D-R18	Kinetics-400	72.8	44.0
Temp Trans† (*Jenni et al.*, 2020)	3D-R18	Kinetics-600	79.3	49.8
RSPNet† (*Chen et al.*, 2021a)	3D-R18	Kinetics-400	74.3	41.8
MFO† (*Qian et al.*, 2021a)	3D-R18	Kinetics-400	79.1	47.6
TimeEq† (*Jenni and Jin*, 2021)	3D-R18	Kinetics-400	87.1	63.6
ASCNet† (*Huang et al.*, 2021)	3D-R18	Kinetics-400	80.5	52.3
TCLR† (*Dave et al.*, 2022)	3D-R18	Kinetics-400	85.4	55.4
DPC (*Han et al.*, 2019)	2D3D-R34	Kinetics-400	75.7	35.7
MemDPC (*Han et al.*, 2020b)	2D3D-R34	Kinetics-400	78.1	41.2
SpeedNet (*Benaim et al.*, 2020)	S3D-G	Kinetics-400	81.1	48.8
CoCLR† (*Han et al.*, 2020a)	S3D-G	Kinetics-400	87.9	54.6
CATE† (*Sun et al.*, 2021)	3D-R50	Kinetics-400	88.4	61.9
CVRL† (*Qian et al.*, 2021b)	3D-R50	Kinetics-400	92.9	67.9
BraVe† (*Recasens et al.*, 2021)	3D-R50	Kinetics-400	93.7	72.0
Random Initialization	2D3D-R18	-	54.4	31.9
Future prediction	2D3D-R18	Kinetics-400	65.9	35.3
BFP (Ours)	2D3D-R18	Kinetics-400	**66.4**	**45.3**

Table 4.2: **Finetuning on UCF101 and HMDB51.** The second block shows methods with different architectures. The third and fourth blocks are methods pretrained by us on UCF101 and Kinetics-400, respectively. For comparison we report results with random initialization in the first block. Future prediction refers to the DPC method trained with our implementation details. While we observe moderate improvements of our method on UCF101, it significantly boosts the performance on HMDB51. † denotes methods that were published concurrently or subsequently to BFP.

Model	spatial negatives	top1 Accuracy on Kinetics-400					
		conv_1	res_1	res_2	res_3	res_4	agg
Random Initialization	-	31.1	26.6	16.9	8.4	2.9	1.8
DPC (*Han et al.*, 2019)	✓	38.1	37.8	27.0	19.3	4.6	3.6
Future Prediction	✓	41.2	40.5	32.8	19.8	4.3	2.4
Ours sobel+crop+rot	✗	41.6	40.5	34.4	26.2	9.8	7.7

Table 4.3: **Layer-wise Evaluations on Kinetics-400.** Pretrained models are frozen up to a convolutional/residual layer `conv`/`res` or completely `agg`, remaining layers are trained from scratch. The augmentations involve random crop applied independently to each partition. Random sobel filtering and rotation with multiplicands of 90° are independently applied to past and future blocks only. Methods with independent augmentations instead of spatial negatives are superior in later layers.

Pretraining on Kinetics-400. Next, we investigate the benefit of large-scale datasets. We pretrain our model on the Kinetics-400 dataset (*Kay et al.*, 2017), and evaluate the learned representations by finetuning on UCF101 and HMDB51. Table 4.2 summarizes the results. We outperform future prediction on both UCF101 and HMDB51, a dataset known to be notoriously difficult for action recognition. We achieve 45.3% top1 accuracy on HMDB51 surpassing many previous self-supervised RGB-only methods by a significant margin. We further compare to various different methods using different network architectures and pretraining datasets in Table 4.2. Note that 2D3D-Resnet34, S3D-G and 3D-Resnet50 are significantly deeper and computationally more expensive than 2D3D-Resnet18. The 3D-Resnet18 architecture is better comparable to ours; here, several more recent methods have been proposed that achieve strong performance surpassing BFP. Among them are the TCLR method (*Dave et al.*, 2022) and TimeEq (*Jenni and Jin*, 2021); both methods also employ a contrastive loss. *Dave et al.* (2022) combine a total of three contrastive losses, including an instance recognition loss, a local-local temporal contrastive loss and a global-local temporal contrastive loss. Both the local-local and the global-local temporal contrastive loss employ a form of temporal hard negatives; here, clips samples from different temporal locations of the same video. When combined with an instance recognition loss, their method achieves strong performance on both UCF101 and HMDB51. This further shows the effectiveness of temporal hard negatives for contrastive video representation learning and the benefit of combining them with an instance recognition loss.

4.3.3 Layer-wise Evaluation on Kinetics-400

Finetuning is a well justified approach in practice. However, being prone to overfitting on the small target dataset, it does not necessarily establish a solid evaluation of the representations. To evaluate the learned representations more elaborately, we transfer the features of a pretrained model for action recognition on the Kinetics-400 dataset. We initialize multiple networks up to different layers with pretrained weights, and train only the remaining layers from scratch while the initialized layers are frozen. Provided enough training data, a better performance indicates higher quality representations of the initialized layers. Table 4.3 and Figure 4.3 show the results when the initialized network is frozen up to the first convolutional (`conv`), 4 residual (`res`) layers, and in the extreme case the aggregation function (`agg`). Moreover, this evaluation allows us to investigate the effect

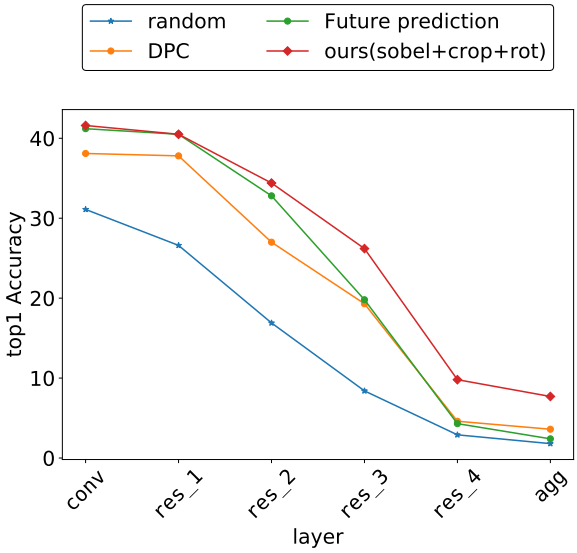


Figure 4.3: **Layer-wise Evaluations on Kinetics-400.** The corresponding results are shown in Table 4.3. The pretrained networks are frozen up to some layer, the remaining are trained from scratch.

of spatial negatives versus independent augmentations. The later layers with larger receptive fields represent global attributes of videos, *e.g.* action class, whereas the earlier layers represent local attributes. We obtain remarkably better performance in the later layers when the network is trained via independent transformations (our approach) compared with spatial negatives (Future prediction). The performance in early layers of both approaches is comparable. This observation confirms our argument in Section 4.2. Spatial negatives result in a representation that is variant with respect to the spatial region of the input and therefore encourage local feature descriptors, which may be useful in early layers where the receptive field is limited. Independent augmentations on the other hand lead to representations that are invariant to spatial transformations, allowing global feature descriptors that are more useful in later layers.

4.3.4 Ablation Studies

Ablation of the Number of Past and Future Clips. Table 4.4 shows the impact of different data partitioning paradigms into past, present, and future blocks. We pretrain different methods on split 1 of UCF101 and report the top1 accuracy obtained via finetuning on the same dataset. We follow the same set up as in Table 4.1. We show each partition with a triplet referring to the number of blocks used to construct past, present, and future blocks respectively. We use all present blocks to construct present features, and randomly select a single block of past and future to build positive pairs of past/future and temporal negatives of future/past. The larger number of future and past blocks allows us to provide the InfoNCE loss with a larger set of positive and negative pairs. For instance, $(2, 4, 2)$ provides 4 positive and negatives pairs per video in a batch. $(3, 2, 3)$ increases the difficulty of the task as more temporal hard negatives will be included in the loss function while the temporal receptive field of the present sequence is reduced. Reducing the number of present blocks

# of (P, V, F)	top1 Accuracy on UCF101
$(2, 4, 2)$	63.6
$(3, 2, 3)$	61.9
$(2, 2, 2)$	**63.8**
$(1, 2, 1)$	62.9
$(2, 1, 2)$	62.5

Table 4.4: **Ablation of the Number of Clips.** Each triplet indicates the number of past, present, and future blocks per video in a batch. All present blocks are used to extract the present features. A pair of randomly taken single past and future blocks are used to construct past/future or future/past features.

while keeping the past and future blocks fixed does not change the quality of the representations significantly. However, further reducing the number of future and past blocks in $(1, 2, 1)$ or the number of present blocks in $(2, 1, 2)$ weakens the representations. The former reduces the number of temporal hard negatives which leads to a simpler task, while the later limits temporal information. This result indicates the effectiveness of temporal hard negatives as well as capability of our model to exploit temporal information.

Ablation of our Architecture. As mentioned above we found that using a different aggregation function (different instance of the ConvGRU) for past and future features while keeping the same 2D3D-Resnet18 backbone achieved better results. Here we want to investigate this further and compare the effect of using the same or different aggregation function or backbone for past and future features, *i.e.* sharing weights of the feature extractor for the present representation z_v and and past/future representation z_{pf}. For this ablation we pretrain our models on UCF101 and evaluate the representations via finetuning. The results are shown in Table 4.5. When using the same aggregation function and the same or different backbone to extract past and future features, we observe a small performance drop compared to the best setting, whereas using both different backbones and different aggregation functions decreases the quality of the representation significantly. We use the best performing setting for all of our experiments.

Backbone f	Aggregation g	top1 Accuracy on UCF101
same	same	62.4
different	same	62.8
same	different	**63.6**
different	different	60.6

Table 4.5: **Ablation of our Architecture.** We investigate the effect of using the *same* or *different* instance of the backbone or aggregation function for present and past/future clips.

Model	temporal negatives	spatial augmentations	top1 Accuracy on UCF101
Random Initialization	-	-	54.4
Ours	✗	-	48.2
Ours	✗	crop	47.3
Ours	✗	crop + flip	47.5
Ours	✗	crop + flip + rot	51.5
Ours	✓	crop + flip	58.2

Table 4.6: **Effectiveness of Temporal Negatives.**

4.3.5 Temporal Negatives

Next, we want to validate the effectiveness of our temporal negatives. We train all methods on UCF101 using spatial transformations that are applied independently to the past, present and future blocks; for evaluation we finetune on UCF101. Note that we do not use spatial negatives in this experiment. All methods in Table 4.6 that do not employ temporal negatives fail – the learned representations are especially bad initializations, performing even worse than random initialization – confirming a degenerate solution via shortcuts. Since neither temporal nor spatial negatives are used here, only random video sequences are considered as negatives and the InfoNCE loss in Eq. (4.1). And while this loss gives a lower bound on the mutual information, plainly optimizing for mutual information does not necessarily lead to useful representations. Adding temporal negatives on the other hand enables our method to learn a representation that is beneficial for action recognition. These experiments confirm the observation in Section 4.2.3 that a structured set of hard negatives which are not sampled from the marginal distributions are more effective for representation learning than an accurate approximation of the mutual information.

4.4 Conclusion

In this chapter we proposed a novel method for self-supervised video representation learning based on bidirectional feature prediction. We showed how both past and future prediction can be jointly incorporated in a contrastive learning framework and demonstrated the effectiveness of our temporal hard negatives via reversing the order of future and past. We validated our design choices empirically and extensively evaluated our method via finetuning and layer-wise evaluations, outperforming unidirectional feature prediction methods on an action recognition downstream task.

While the hard temporal negatives are essential to our method to prevent shortcuts (see Table 4.6), they impose some limitations on the learned representations: First, incorporating temporal negatives based on the reversed order of future and past may result in a representation that mainly focuses on features that help the model to distinguish between future and past, *i.e.* temporal variations, and may focus less on global video-level features that remain the same throughout past, present and future frames, as they do not enable the model to distinguish between future and past. We address this issue in Chapter 6 where we develop a method to capture both stationary (video-level) features and non-stationary ones representing temporal variations. Second, one can argue that such temporal negatives

should not be *true* negatives – ultimately these samples are still highly related to the present frames. Forcing the model to embed present and reverse future-past samples far apart in the feature space may be sub-optimal. Instead, these samples should be treated in a more nuanced fashion. Unfortunately, the standard contrastive learning framework requires a strict binary separation into positive and negative pairs and impedes a more sophisticated treatment of more or less related samples. To alleviate this restriction, we introduce a method in Chapter 5 to incorporate a ranked set of positive pairs that allows the user to define the desired similarities and relevance of samples.

CHAPTER 5

Ranking InfoNCE: Boosting Contrastive Learning via Ranked Positives

In Chapter 4, we have developed a contrastive video representation learning method that incorporates temporal hard negatives to learn temporally structured representations. While these samples are very effective for learning video representations in practice, it is unclear whether they should be considered true negatives since they are still highly related to the present frames. In general, the strict binary separation into positive and negative pairs in contrastive learning may not always be possible; in this chapter we propose a method to properly treat samples that lie in-between. For another example related to video representation learning, consider temporal crops of videos, *i.e.* clips taken at different times in the video: Treating all temporal crops as positives encourages representations that are invariant to the changes over time, while treating them as negatives guides the model to ignore high-level global features that are shared between temporal crops, such as the video-level action class. In reality, the content of videos typically changes gradually over time, which should be mirrored in the feature space. To properly incorporate such samples, we propose a novel ranking InfoNCE loss (RINCE) that not only allows for but also benefits from a ranked definition of positives. Essentially, this loss consists of a series of InfoNCE losses, one for each rank, where higher-ranked positives incorporate lower-ranked positives as hard negatives. We evaluate our approach extensively under various settings with varying levels of supervision: A fully supervised setting with ground truth ranking allows us to investigate the method without confounding noise, then, we demonstrate the benefit of large-scale training with noisy similarities in place of ground truth rankings, and finally, we apply RINCE to the fully unsupervised setting of video representation learning.

Individual Contribution

The following chapter is based on the publication (*Hoffmann et al.*, 2022):

Ranking Info Noise Contrastive Estimation: Boosting Contrastive Learning via Ranked Positives
David Hoffmann*, Nadine Behrmann*, Juergen Gall, Thomas Brox, and Mehdi Noroozi
The AAAI Conference on Artificial Intelligence (AAAI), 2022.
doi: 10.1609/aaai.v36i1.19972.
* David Hoffmann and Nadine Behrmann are joint first authors.

This publication was done in very close collaboration between David Hoffmann, Mehdi Noroozi and Nadine Behrmann. Juergen Gall and Thomax Brox provided scientific guidance and very valuable feedback and suggestions. The initial idea to apply a series of InfoNCE losses to enforce

gradually decreasing similarities of ranked positives was proposed by Mehdi Noroozi. The different RINCE variants were jointly developed by David Hoffmann, Mehdi Noroozi and Nadine Behrmann. The idea to rank different clips of the same video based on their temporal distance was developed by Mehdi Noroozi and Nadine Behrmann and implemented by Nadine Behrmann. While Nadine Behrmann proposed to exploit label information to construct ranked positives, the idea to use *other* classes as lower ranked positives was proposed by David Hoffmann, both using superclasses and RoBERTa class similarity scores. The experiments were implemented by David Hoffmann and Nadine Behrmann.

The retrieval evaluation was proposed in discussions with David Hoffmann, Mehdi Noroozi and Nadine Behrmann and implemented by Nadine Behrmann; evaluating the retrieval performance via mAP was suggested by Thomas Brox. The evaluation via out-of-distribution detection was suggested in discussions between David Hoffmann, Mehdi Noroozi and Thomas Brox and implemented by David Hoffmann. Visualizing the embedding spaces via t-SNE plots was suggested and implemented by David Hoffmann. The idea to visualize the class similarity matrix was jointly developed by David Hoffmann, Mehdi Noroozi and Nadine Behrmann and implemented by Nadine Behrmann. The RINCE loss analysis was carried out by David Hoffmann and Mehdi Noroozi. The analysis on the role of τ was jointly done by David Hoffmann and Nadine Behrmann.

Contents

5.1 Introduction

Contrastive learning recently triggered progress in self-supervised representation learning. Most existing variants require a strict definition of positive and negative pairs used in the InfoNCE loss or simply ignore samples that can not be clearly classified as either one or the other (*Zhao et al.*, 2021). Contrastive learning forces the network to impose a similar structure in the feature space by pulling the positive pairs closer to each other while keeping the negatives apart.

This binary separation into positives and negatives can be limiting whenever the boundary between those is blurry. For example, different samples from the same classes are used as negatives

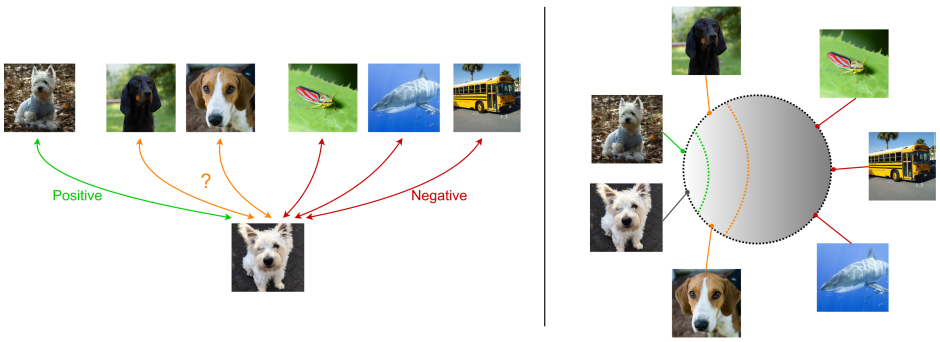


Figure 5.1: **Contrastive Learning Should not Be Binary.** In many scenarios a strict separation of samples in "positives" and "negatives" is not possible. So far, this grey zone (left) was neglected, leading to sub-optimal results. We propose a solution to this problem, which embeds same samples very close and similar samples close in the embedding space (right).

for *instance recognition*, which prevents the network from exploiting their similarities. One way to address this issue is supervised contrastive learning (SCL) (*Khosla et al.*, 2020), which takes class labels into account when making pairs: samples from the same class are treated as positives, while samples of different classes pose negatives. However, even in this optimal setting with ground truth labels, the problem persists – semantically similar classes share many visual features (*Deselaers and Ferrari*, 2011) with the query – and some samples cannot clearly be categorized as either positive or negative, *e.g.* the dog breeds in Fig. 5.1. Treating them as positives makes the network invariant towards the distinct attributes of the samples. As a result, the network struggles to distinguish between different dog breeds. If they are treated as negatives, the network cannot exploit their similarities. For transfer learning to other tasks, *e.g.* out-of-distribution detection, a clean structure of the embedding space, such that samples sharing certain attributes will be closer, is beneficial.

Another example comes from video representation learning: In addition to spatial crops as for images, videos allow to create temporal crops, *i.e.* creating a sample from different frames of the same video. To date, it is an open point of discussion whether temporally different clips from the same video should be treated as positive (*Feichtenhofer et al.*, 2021) or negative (*Dave et al.*, 2022). Treating them as positives will force the network to be invariant towards changes over time, but treating them as negatives will encourage the network to ignore the features that stay constant. In summary, a binary classification in positive and negative will, for most applications, lead to a sub-optimal solution. To the best of our knowledge, a method that benefits from a fine-grained definition of negatives, positives and various states in between is missing.

As a remedy, we propose Ranking Info Noise Contrastive Estimation (RINCE). RINCE supports a fine-grained definition of negatives and positives. Thus, models trained with RINCE can take advantage of various kinds of similarity measures. For example, similarity measures can be based on class similarities, gradual changes of content within videos, pretrained feature embeddings, or even the camera positions in a multi-view setting, etc. In this work, we demonstrate the effectiveness of RINCE with rankings from class similarities and gradual changes in videos.

RINCE puts higher emphasis on similarities between related samples than SCL and cross-entropy, resulting in a richer representation. We show that RINCE learns to represent semantic similarities in the embedding space, such that more similar samples are closer than less similar sam-

ples. Key to this is a new InfoNCE-based loss, which enforces gradually decreasing similarity with increasing rank of the samples. More specifically, our proposed loss consists of a series of InfoNCE losses, one for each rank, where lower ranked positives are incorporated as hard negatives in the InfoNCE loss for higher ranked positives.

The representation learned with RINCE on Cifar-100 improves significantly over cross-entropy for classification, retrieval and OOD detection, and outperforms the stronger SCL baseline (*Khosla et al.*, 2020). Here, improvements are particularly large for retrieval and OOD detection. To obtain ranked positives for RINCE, we use the superclasses of Cifar-100. Furthermore, we demonstrate that RINCE works on large scale datasets and in more general applications, where ranking of samples is not initially given and contains noise. To this end, we show that RINCE outperforms our baselines on ImageNet-100 using only noisy ranks provided by an off-the-shelf natural language processing model (*Liu et al.*, 2019). Finally, we showcase that RINCE can be applied to the fully unsupervised setting, by training RINCE unsupervised on videos, treating temporally far clips as lower ranked positives. This results in a higher accuracy on the downstream task of video action classification then our baselines and even outperforms recent video representation learning methods.

In summary, our contributions are: 1) We propose a new InfoNCE-based loss that replaces the binary definition of positives and negatives by a ranked definition of similarity. 2) We study the properties of RINCE in a controlled supervised setting. Here, we show mild improvements on Cifar-100 classification and sensible improvements for OOD detection. 3) We show that RINCE can handle significant noise in the similarity scores and leads to improvements on large scale datasets. 4) We demonstrate the applicability of RINCE to self-supervised learning with noisy similarities in a video representation learning task and show improvements over InfoNCE in all downstream tasks. 5) Code is available at[1].

5.2 Method

Before introducing our ranking InfoNCE loss, we will recapitulate the contrastive losses introduced in Section 3.1.3 and their characteristics and use cases as these losses form the basis for our ranking InfoNCE.

5.2.1 InfoNCE

We start with the most basic form of the InfoNCE loss. In this setting, two different views of the same data – *e.g.* two different augmentations of the same image – are pulled together in feature space, while pushing views of different samples apart. More specifically, for a query q, a single positive p and a set of negatives $\mathcal{N} = \{n_1, \dots n_k\}$ is given. The views are fed to an encoder network f, followed by a projection head h (*Chen et al.*, 2020b). To measure the similarity between a pair of features we use the cosine similarity sim, see Eq. (3.4). Overall, the task is to train a critic $\varphi(x, y) = \text{sim}\left(h(f(x)), h(f(y))\right)$ using the loss:

$$\mathcal{L} = -\log \frac{\exp\left(\frac{\varphi(q,p)}{\tau}\right)}{\exp\left(\frac{\varphi(q,p)}{\tau}\right) + \sum_{n\in\mathcal{N}} \exp\left(\frac{\varphi(q,n)}{\tau}\right)}, \tag{5.1}$$

[1] https://github.com/boschresearch/rince

where τ is a temperature parameter (*Chen et al.*, 2020b). The above loss relies on the assumption that a single positive pair is available. One drawback with this approach is that all other samples are treated as negatives, even if they are semantically close to the query. Potential solutions include removing them from the negatives (*Zhao et al.*, 2021) or adding them to the positives (*Khosla et al.*, 2020), which we denote by $\mathcal{P} = \{p_1, \dots, p_l\}$. In other cases, we naturally have access to more than one positive, *e.g.* we can sample several clips from a single video, see Figure 5.5. Having multiple positives per query leaves two options, which we discuss in the following.

Log_{out} Positives. A straightforward approach to include multiple positives is to compute Eq. (5.1) for each of them, *i.e.* take the sum over positives outside of the log. This enforces similarity between all positives during training, which suits a clean set of positives well.

$$\mathcal{L}^{\text{out}} = -\sum_{p\in\mathcal{P}} \log \frac{\exp\left(\frac{\varphi(q,p)}{\tau}\right)}{\exp\left(\frac{\varphi(q,p)}{\tau}\right) + \sum_{n\in\mathcal{N}} \exp\left(\frac{\varphi(q,n)}{\tau}\right)}. \tag{5.2}$$

However, the set of positives can be noisy, *e.g.* sampling a temporally distant clip may include sub-optimal positives due to drastic changes in the video.

Log_{in} Positives. An alternative approach, which is more robust to noise or inaccurate samples (*Miech et al.*, 2020), is to take the sum inside the log, Eq. (5.3). To minimize this loss, the network is not forced to set a high similarity to all pairs. It can neglect the noisy/false positives, given that a sufficiently large similarity is set for the true positives, see Table 5.6. However, if a discrepancy between positives exists, it results in a degenerate solution of discarding hard positives. For instance, consider supervised learning where both augmentations and class positives are available for a given query: the class positives, which are harder to optimize, can be ignored.

$$\mathcal{L}^{\text{in}} = -\log \frac{\sum_{p\in\mathcal{P}} \exp\left(\frac{\varphi(q,p)}{\tau}\right)}{\sum_{p\in\mathcal{P}} \exp\left(\frac{\varphi(q,p)}{\tau}\right) + \sum_{n\in\mathcal{N}} \exp\left(\frac{\varphi(q,n)}{\tau}\right)}. \tag{5.3}$$

The above methods assume a binary set of positives and negatives. Thus, they can not exploit the similarity of positives and negatives. In the following, we discuss the proposed ranking version of InfoNCE that allows us to preserve the order of the positives and benefit from the additional information.

5.2.2 RINCE: Ranking InfoNCE

Let us assume that for a given query sample q, we have access to a set of ranked positives in a form of $\mathcal{P}_1, \dots, \mathcal{P}_r$, where $\mathcal{P}_i$ includes the positives of rank i. Let us also assume $\mathcal{N}$ is a set of negatives. Our objective is to train a critic φ such that:

$$\varphi(q, p_1) > \dots > \varphi(q, p_r) > \varphi(q, n) \quad \text{for all } p_i \in \mathcal{P}_i, n \in \mathcal{N}. \tag{5.4}$$

Note that $\mathcal{P}_i$ can contain multiple positives. For ease of notation we omit these indices. To impose the desired ranking presented by the positive sets, we use InfoNCE in a recursive manner where we

start with the first set of positives, treat the remaining positives as negatives, drop the current positive, and move to the next. We repeat this procedure until there are no positives left. More precisely, the loss function reads $\mathcal{L}_{\text{rank}} = \sum_{i=1}^{r} \ell_i$, where

$$\ell_i = -\log \frac{\sum\limits_{p \in \mathcal{P}_i} \exp\left(\frac{\varphi(q,p)}{\tau_i}\right)}{\sum\limits_{p \in \bigcup_{j \geqslant i} \mathcal{P}_j} \exp\left(\frac{\varphi(q,p)}{\tau_i}\right) + \sum\limits_{n \in \mathcal{N}} \exp\left(\frac{\varphi(q,n)}{\tau_i}\right)} \tag{5.5}$$

and $\tau_i < \tau_{i+1}$. Eq. (5.5) denotes the $\mathcal{L}^{\text{in}}$ version of InfoNCE for positives of same rank; other variants are summarized in Table 5.1. The rational behind this loss is simple: The i-th loss is optimized when for all i, j, n the following three conditions are met

1. $\exp(\varphi(q, p_i)/\tau_i) \gg 0$,
2. $\exp(\varphi(q, p_j)/\tau_i) \to 0$ for $j > i$,
3. $\exp(\varphi(q, n)/\tau_i) \to 0$.

Requirements 1 and 2 are competing across the losses: ℓ_i entails $\exp(\varphi(q, p_{i+1})/\tau_i) \to 0$ but ℓ_{i+1} requires $\exp(\varphi(q, p_{i+1})/\tau_{i+1}) \gg 0$. This requires the model to trade-off the respective loss terms, resulting in a ranking of positives $\varphi(q, p_i) > \varphi(q, p_{i+1})$. Requirement 3 ensures $\varphi(q, p_i) > \varphi(q, n)$. In combination these conditions achieve the desired ranking in Eq. 5.4.

In the following we explain the intuition behind our choice of τ values based on the analyses of (*Wang and Liu*, 2021); for a more detailed analysis see Section 5.2.4. A low temperature in the InfoNCE loss results in a larger relative penalty on the high similarity regions, *i.e.* hard negatives. As the temperature increases, the relative penalty distributes more uniformly, penalizing all negatives equally. A low temperature in ℓ_i allows the network to concentrate on forcing $\varphi(q, p_i) > \varphi(q, p_{i+1})$, ignoring easy negatives. A higher temperature on ℓ_r relaxes the relative penalty of negatives with respect to p_r so that the network can enforce $\varphi(q, p_r) > \varphi(q, n)$.

5.2.3 RINCE Loss Variants

The version in Eq. (5.5) corresponds to the $\mathcal{L}_{\text{in}}$ variant of InfoNCE. We summarize other variants in Table 5.1 and spell them out in more detail in the following.

Naming	# Positives per Rank	Loss
RINCE-uni	single	Eq. (5.1)
RINCE-out	multiple	Eq. (5.2)
RINCE-in	multiple	Eq. (5.3)
RINCE-out-in	multiple	Eq. (5.2) (ℓ_1); Eq. (5.3) ($\ell_i, i > 1$)

Table 5.1: **Different Variants of RINCE.** For the exact loss functions we refer to the text.

RINCE-uni. In the RINCE-uni loss variant only a single positive is given for each rank, *i.e.* $\mathcal{P}_i = \{p_i\}$. In this case, we use Eq. (5.1) for the individual loss terms (Eq. (5.2) and Eq. (5.3) are the same for a single positive), and the loss reads:

$$\mathcal{L}_{\text{RINCE-uni}} = -\sum_{i=1}^{r} \log \frac{\exp(\varphi(q, p_i)/\tau_i)}{\sum_{j=i}^{r} \exp(\varphi(q, p_j)/\tau_i) + \sum_{n\in\mathcal{N}} \exp(\varphi(q, n)/\tau_i)}. \tag{5.6}$$

RINCE-out. For the RINCE-out loss variant we take the sum over positives outside of the log for each rank i:

$$\mathcal{L}_{\text{RINCE-out}} = -\sum_{i=1}^{r} \sum_{p\in\mathcal{P}_i} \log \frac{\exp(\varphi(q, p)/\tau_i)}{\sum_{p\in\{p_i\}\cup(\bigcup_{j>i}\mathcal{P}_j)} \exp(\varphi(q, p)/\tau_i) + \sum_{n\in\mathcal{N}} \exp(\varphi(q, n)/\tau_i)}. \tag{5.7}$$

RINCE-in. On the other hand, we take the sum over positives inside of the log for the RINCE-in loss variant:

$$\mathcal{L}_{\text{RINCE-in}} = -\sum_{i=1}^{r} \log \frac{\sum_{p\in\mathcal{P}_i} \exp(\varphi(q, p)/\tau_i)}{\sum_{p\in\bigcup_{j\geqslant i}\mathcal{P}_j} \exp(\varphi(q, p)/\tau_i) + \sum_{n\in\mathcal{N}} \exp(\varphi(q, n)/\tau_i)}. \tag{5.8}$$

Note that there is a subtle but noteworthy difference to the log-in version of (*Khosla et al.*, 2020), who compute the mean rather than the sum inside the log. As they observe a significantly worse performance of their log-in version, we decide to use the version proposed in (*Miech et al.*, 2020; *Han et al.*, 2020a) for all our experiments, including the baseline SCL-in.

RINCE-out-in. Finally, we consider a combination of the two above: Whenever noise for first rank positives can be expected to be low, while it might be higher for higher rank positives we can use the out-option for the first rank and the in-option for the remaining ranks:

$$\begin{aligned}\mathcal{L}_{\text{RINCE-out-in}} = &-\sum_{p\in\mathcal{P}_1} \log \frac{\exp(\varphi(q, p)/\tau_1)}{\sum_{p\in\{p_i\}\cup(\bigcup_{j>1}\mathcal{P}_j)} \exp(\varphi(q, p)/\tau_1) + \sum_{n\in\mathcal{N}} \exp(\varphi(q, n)/\tau_1)} \\ &-\sum_{i=2}^{r} \log \frac{\sum_{p\in\mathcal{P}_i} \exp(\varphi(q, p)/\tau_i)}{\sum_{p\in\bigcup_{j\geqslant i}\mathcal{P}_j} \exp(\varphi(q, p)/\tau_i) + \sum_{n\in\mathcal{N}} \exp(\varphi(q, n)/\tau_i)}.\end{aligned} \tag{5.9}$$

5.2.4 RINCE Loss Analysis

In the following we will give a more theoretical analysis of RINCE, explain why it leads to ranking and justify our choice of setting $\tau_i < \tau_{i+1}$. First, we will study the relative penalty (*Wang and Liu*, 2021) to identify which negatives contribute most to the individual RINCE terms, dependent on the choice of τ and elaborate how this results in ranking. Next, we will elaborate how the choice of $\tau_i < \tau_{i+1}$ guides the network towards a desired trade-off between the opposing terms in the RINCE loss.

Which loss term focuses on which negatives? Following (*Wang and Liu*, 2021), we investigate the impact of negatives in the loss relative to the impact of positives, which is referred to as *relative penalty*. The relative penalty r_n^p is obtained by dividing the gradient magnitude with respect to the similarity of negative and query ($s_{q,n} = \varphi(q,n)$) by the gradient magnitude with respect to the similarity of positive and query ($s_{q,p} = \varphi(q,p)$):

$$r_n^p = \left|\frac{\partial \ell(q)}{\partial s_{q,n}}\right| \Big/ \left|\frac{\partial \ell(q)}{\partial s_{q,p}}\right|. \tag{5.10}$$

A large relative penalty for a given negative n implies a large contribution of this negative in the InfoNCE loss. For simplicity, we consider only two ranks but extending the explanation to more ranks is straightforward. Let $p_1 \in \mathcal{P}_1$ denote a first rank positive, $p_2 \in \mathcal{P}_2$ a second rank positive and $n \in \mathcal{N}$ a negative. The relative penalty of a negative $n \in \mathcal{N}$ with respect to p_2 in ℓ_2 is given by:

$$r_n^{p_2} = \frac{\exp(\varphi(q,n)/\tau_2)}{\sum\limits_{x \in \mathcal{N}\setminus\{n\}} \exp(\varphi(q,x)/\tau_2)} \tag{5.11}$$

Similarly, the relative penalty of n with respect to p_1 in ℓ_1 is:

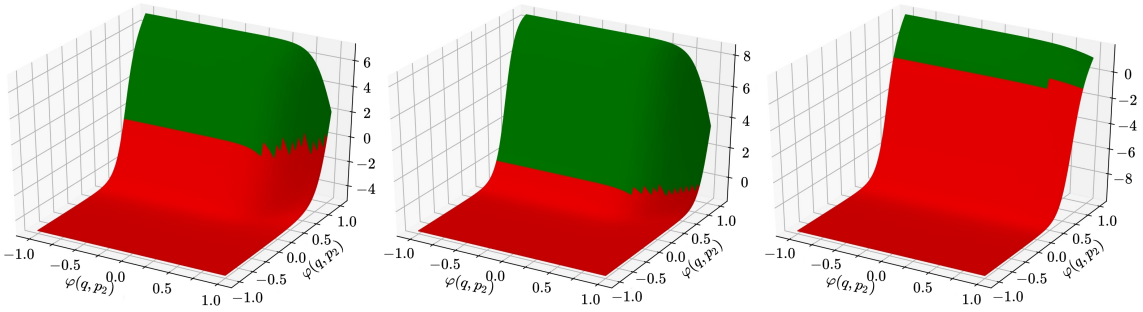
$$r_n^{p_1} = \frac{\exp(\varphi(q,n)/\tau_1)}{\sum\limits_{x \in \mathcal{N}\setminus\{n\}} \exp(\varphi(q,x)/\tau_1) + \sum\limits_{p_2^* \in \mathcal{P}_2} \exp(\varphi(q,p_2^*)/\tau_1)} \tag{5.12}$$

and for p_2 with respect to p_1 in ℓ_1:

$$r_{p_2}^{p_1} = \frac{\exp(\varphi(q,p_2)/\tau_1)}{\sum\limits_{x \in \mathcal{N}} \exp(\varphi(q,x)/\tau_1) + \sum\limits_{p_2^* \in \mathcal{P}_2\setminus\{p_2\}} \exp(\varphi(q,p_2^*)/\tau_1)}. \tag{5.13}$$

Note that p_2 serves as a negative for p_1 in ℓ_1 and a positive in ℓ_2. With small τ_1 Eq. (5.12) is larger for close samples of n than with larger τ_1. Therefore, small τ_1 result in significantly larger relative penalties for n close to q, in comparison to a larger τ_1. Both also hold for Eq. (5.13) and p_2. Thus, increasing τ shifts the focus of the loss function from close negatives to a more uniform contribution of all negatives. With $\tau_2 > \tau_1$, Eq. (5.11) is more uniform over different similarity scores than Eq. (5.12) and (5.13) (compare Figure 3 in (*Wang and Liu*, 2021)). Since $\varphi(q,p_2) > \varphi(q,n)$ is enforced by the positive "pull force" in ℓ_2, we have $r_{p_2}^{p_1} \gg r_n^{p_1}$. Therefore, higher emphasize is put on $\varphi(q,p_1) > \varphi(q,p_2)$ than on $\varphi(q,p_1) > \varphi(q,n)$ in ℓ_1 and, intuitively, ℓ_2 emphasizes $\varphi(q,p_2) > \varphi(q,n)$. Thus, ℓ_1 ensures that p_1 and p_2 can be discriminated well and ℓ_2 ensures discrimination between p_2 and n. In other words, with increasing rank, and thus increasing τ, the focus of the loss gradually shifts from close negatives towards all negatives, effectively increasing with each rank the radius around q at which significant relative penalty results from the negatives. This "pushing" force with gradual increasing radius from the respective negatives in combination with the pulling of the respective positives towards q results in ranking.

How τ influences the optimal solution, controls the trade-off between opposing loss terms and why $\tau_i < \tau_{i+1}$ is a good choice for ranking. To answer these question we study the trade-off mechanism between negatives in ℓ_1 and the positives in ℓ_2, *i.e.* the opposing terms in the RINCE loss



(a) $\tau_1 = 0.1, \tau_2 = 0.2$ (b) $\tau_1 = 0.1, \tau_2 = 0.7$ (c) $\tau_1 = 0.2, \tau_2 = 0.1$

Figure 5.2: **Trade-off Between Opposing RINCE Terms for Varying Values of τ_1 and τ_2.** We visualize $\left|\frac{\partial \ell_1}{\partial p_2}\right| - \left|\frac{\partial \ell_2}{\partial p_2}\right|$ for different values of $\varphi(q, p_1)$ and $\varphi(q, p_2)$ ($\varphi(q, n)$ is fixed). We show plots for different τ values. Red indicates negative values and green positive values. In the red region $\varphi(q, p_2)$ is maximized in the green region minimized. The equilibrium for a fixed $\varphi(q, p_1)$ lies on the boundary of red and green.

that lead to the ranking. To investigate the trade-off, we compare the gradient with respect to the negatives p_2 in ℓ_1 with the gradients with respect to the positives in ℓ_2 (also p_2). We want to study which term dominates the entire gradient under which conditions. For this purpose we define

$$K(\varphi(q, p_1), \varphi(q, p_2)) = \left|\frac{\partial \ell_1}{\partial p_2}\right| - \left|\frac{\partial \ell_2}{\partial p_2}\right|, \tag{5.14}$$

which is given by

$$\begin{aligned} K(\varphi(q, p_1), \varphi(q, p_2)) = &\frac{1}{\tau_1} \frac{\exp(\varphi(q, p_2)/\tau_1)}{\sum\limits_{n \in \mathcal{N} \cup \mathcal{P}_2} \exp(\varphi(q, n)/\tau_1) + \exp(\varphi(q, p_1)/\tau_1)} \\ &- \frac{1}{\tau_2} \frac{\sum_{n \in \mathcal{N}} \exp(\varphi(q, n)/\tau_2)}{\sum\limits_{n \in \mathcal{N}} \exp(\varphi(q, n)/\tau_2) + \exp(\varphi(q, p_2)/\tau_2)} \end{aligned} \tag{5.15}$$

Intuitively, the value of $K(\varphi(q, p_1), \varphi(q, p_2))$ in Eq. (5.14) shows which term dominates the gradient. In our case it even holds that $K(\varphi(q, p_1), \varphi(q, p_2)) = \left|\frac{\partial \ell_1}{\partial p_2}\right| - \left|\frac{\partial \ell_2}{\partial p_2}\right| = \frac{\partial \ell_1}{\partial p_2} + \frac{\partial \ell_2}{\partial p_2}$, thus, $K(\varphi(q, p_1), \varphi(q, p_2))$ corresponds to the actual sum of the two gradients, meaning that $K(\varphi(q, p_1), \varphi(q, p_2)) = 0$ means the gradients cancel each other out. There exist three different cases:

- $K > 0$: $\left|\frac{\partial \ell_1}{\partial p_2}\right| > \left|\frac{\partial \ell_2}{\partial p_2}\right|$, *i.e.* $\frac{\partial \ell_1}{\partial p_2}$ dominates the gradient and effectively ℓ_1 minimizes $\varphi(q, p_2)$.
- $K < 0$: $\left|\frac{\partial \ell_1}{\partial p_2}\right| < \left|\frac{\partial \ell_2}{\partial p_2}\right|$, *i.e.* $\frac{\partial \ell_2}{\partial p_2}$ dominates the gradient and effectively ℓ_2 maximizes $\varphi(q, p_2)$.
- $K = 0$: An equilibrium between the opposing terms in ℓ_1 and ℓ_2 is found – neither $\frac{\partial \ell_1}{\partial p_2}$ nor $\frac{\partial \ell_2}{\partial p_2}$ dominates the gradient.

We visualize Eq. (5.14) in Figure 5.2. To this end, we model $\varphi(q, n)$ with a Gaussian with $\mu = 0.1$ and $\sigma = 0.1$ and show different combinations of τ values, namely $\tau_1 = 0.1, \tau_2 = 0.2$ in Figure 5.2a,

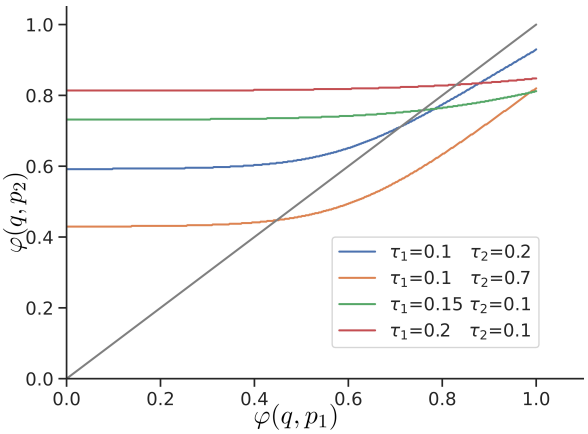


Figure 5.3: **Equilibrium Lines: Solutions for $K(\varphi(q, p_1), \varphi(q, p_2)) = 0$ for different values of τ_1 and τ_2.** The lines show all combinations of $\varphi(q, p_1)$ and $\varphi(q, p_2)$ that are solutions of the opposing RINCE terms, *i.e.* the gradients are identical except for the sign. The grey line indicates $\varphi(q, p_1) = \varphi(q, p_2)$. For $\tau_1 > \tau_2$ it can be seen that with increasing τ_1, the curvature of the equilibrium line steadily decreases resulting in a almost constant value for $\varphi(q, p_2)$ for $\tau_2 = 0.2$.

$\tau_1 = 0.1, \tau_2 = 0.7$ in Figure 5.2b and $\tau_1 = 0.2, \tau_2 = 0.1$ in Figure 5.2c. For convenience we color negative values in red and positive values in green. The *equilibrium line* separates red and green area. For easier comparison of the equilibrium lines, we visualize them in Figure 5.3. It shows the value of $\varphi(q, p_2)$ that results in an equilibrium state as a function of $\varphi(q, p_1)$, *i.e.* it shows the projection of the equilibrium lines of Figure 5.2 to the $\varphi(q, p_1), \varphi(q, p_2)$ plane. We make the following observations:

1. For small values of $\varphi(q, p_1)$ the equilibrium line is close to constant – the optimization behavior of $\varphi(q, p_2)$ is not influenced by the values of $\varphi(q, p_1)$ – however, this only occurs very early in training, as $\varphi(q, p_1)$ is maximized without any opposing loss terms.

2. When $\tau_1 < \tau_2$, the equilibrium line indicates that $\varphi(q, p_2)$ grows proportionately to $\varphi(q, p_1)$ (see Figure 5.3 blue and orange).

3. When $\tau_1 < \tau_2$, $\varphi(q, p_1) > \varphi(q, p_2)$ for the interesting regions (see grey line in Figure 5.3). τ_2 can be used to control the minimal similarity of $\varphi(q, p_1)$ required for ranking to appear. Thus, higher ranks require larger τ_2.

4. When increasing τ_2 for a fixed τ_1, the equilibrium line drops to smaller values overall – the trade-off is achieved with a smaller similarity of $\varphi(q, p_2)$ (compare Figure 5.3). Again, this suits higher ranks better.

5. When $\tau_1 > \tau_2$ the curvature of the equilibrium line decreases steadily. For example, with $\tau_1 = 0.2$ and $\tau_2 = 0.1$ it is almost constant. The optimization of $\varphi(q, p_2)$ is barely influenced by the values of $\varphi(q, p_1)$ (see green and red line in Figure 5.3).

Aside from our theoretical justification, we empirically demonstrate that our loss preserves the desired ranking in the feature space, see Figure 5.7.

5.3 Experiments

We first study the properties of RINCE in the controlled supervised setting on Cifar-100 in Section 5.3.2, where we evaluate the model via classification accuracy and retrieval. To investigate our learned representation beyond standard metrics, we further evaluate our model's ability for out-of-distribution (OOD) detection. In Section 5.3.4, we show that RINCE leads to significant improvements on the large scale dataset ImageNet-100 in terms of accuracy and OOD, even with more noisy similarity scores. Last, in Section 5.3.5, we demonstrate the effectiveness of RINCE in a fully unsupervised setting, namely, on an unsupervised video representation learning task. For all experiments we follow the MoCo v2 setting (*Chen et al.*, 2020d) with a momentum encoder, a memory bank and a projection head. Throughout the section, we compare different versions of RINCE (Table 5.1), to study their behavior in different settings.

5.3.1 Implementation Details

First, we describe the implementation details for the methods trained in Section 5.3.2 and 5.3.4; the implementation details for the video representation learning experiments can be found in Section 5.3.5. Our experiments are based on the Pytorch implementation of (*Khosla et al.*, 2020). Common hyper-parameters are used, as given by (*Khosla et al.*, 2020). Namely, we use ResNet-50 for all models trained on Cifar-100. For ImageNet-100 we use ResNet-18. During contrastive training we use a projection head with a single hidden layer with dimension of 2048 and an output dimension of 128. The projection head is discarded after the contrastive learning stage.

Optimizer. For both, SCL and RINCE we use stochastic gradient descent with a learning rate of 0.5, batch size of 512, momentum of 0.9, weight decay of $1\mathrm{e}-4$ and a cosine learning rate scheduler. All models are trained for 1000 epochs. 500 epochs lead to slightly worse results and results do not change significantly when training for 2000 instead of 1000 epochs. Baselines using cross-entropy loss are trained only for 500 epochs, as we observed that accuracy decreases after epoch 500. For cross-entropy we use a learning rate of 0.8, following (*Khosla et al.*, 2020). In contrast to *Khosla et al.* (2020) we use a batch size of 512. We also tested the square root scaling rule for the learning rate, as proposed in (*Krizhevsky*, 2014), but achieve lower accuracy. Our accuracy matches the one reported by *Khosla et al.* (2020), despite the smaller batch size.

For ImageNet-100 we use a batch size of 768 and find a learning rate of 0.3 to give best results for RINCE and cross-entropy. For SCL we find 0.325 to give best results. We train all models for 500 epochs.

Data Augmentation. We use the same set of standard data augmentations as *Khosla et al.* (2020) in their Pytorch implementation. We create a random crop of size between 20% to 100% of the initial image with random aspect ration between $3/4$ and $4/3$ of the initial aspect ratio. The resulting crop is scaled to 32×32 pixels for Cifar-100 and 224×224 for ImageNet-100. We flip images with a probability of 0.5 and apply color jitter randomly selected from $[0.6, 1.4]$ for brightness, contrast and saturation and apply jitter to the hue from $[-0.1, 0.1]$ with a probability of 0.8. Finally, we convert the image to grayscale with probability of 0.2. Cross-entropy does not use color jitter and random

grayscaling, but except from that uses the same augmentations. Cross-entropy strong augmentation (cross-entropy s.a.) uses the exact same augmentations as RINCE and SCL.

Computational Cost. RINCE only adds a small computational cost to the training pipeline, as only $r-1$ additional computations of the InfoNCE loss function (Eq. (5.1)) are necessary. Note, that the dot products of q and all $p \in \mathcal{P}$ have to be computed only once and the respective results can be reused for each rank specific loss ℓ_i (Eq. (5.5)). In our experiments we did not observe a noticeable difference in overall training time between RINCE and SCL.

5.3.2 Learning from Class Hierarchies

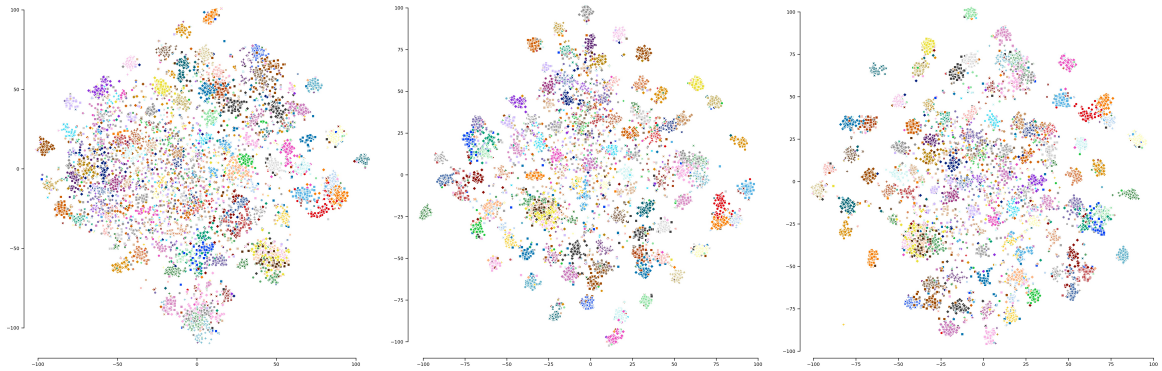
The optimal testbed to study the proposed loss functions is the supervised contrastive learning (SCL) setting. The effect of the proposed loss functions can be studied without confounding noise, using ground truth labels and ground truth rankings. The SCL method of (*Khosla et al.*, 2020) considers all samples with the same class as positives; thus, either Eq. (5.2) or Eq. (5.3) is used. However, semantically similar classes share similar visual features (*Deselaers and Ferrari*, 2011). When strictly treated as negatives the model does not mirror the structure available by the labels in its feature space, although, this is favorable for transferability to other tasks. RINCE allows the model to keep this structure, and learn not only dissimilarities between, but also similarities across classes. We show quantitatively that RINCE learns a higher quality representation than cross-entropy and SCL on Cifar-100 and ImageNet-100 by evaluating on linear classification, image retrieval, and OOD tasks. Unless otherwise stated, we report results for ResNet-50.

Datasets. **Cifar-100** (*Krizhevsky et al.*, 2009) provides a semantic hierarchy, which in addition to class labels also contains superclass labels. We use this hierarchy to define first rank positives (same class) and second rank positives (same superclass). **TinyImageNet** (*Le and Yang*, 2015) is a small version of ImageNet (*Deng et al.*, 2009) comprising only 200 of the 1000 classes, each consisting of 500 samples at 64×64 pixel resolution. **ImageNet-100** (*Tian et al.*, 2020a) is a subset of ImageNet, consisting of 100 classes of ImageNet at full resolution. We use the RoBERTa (*Liu et al.*, 2019) model to obtain semantic word embeddings for all class names in ImageNet-100. Second rank positives are based on the word embedding similarity and a predefined threshold.

Baselines. We compare our RINCE method to a variety of different baselines, including **cross-entropy**, cross-entropy with the same augmentations as RINCE (**cross-entropy s.a.**), **Triplet** loss (*Weinberger et al.*, 2006) and **SCL** (*Khosla et al.*, 2020), trained with Eq. (5.2) (**SCL-out**) or Eq. (5.3) (**SCL-in**). An advantage of RINCE compared to these baselines is that it benefits from extra information provided by the superclasses. To show that making use of this knowledge is not trivial, we further compare RINCE to the following baselines that leverage superclass information in different ways: 1) **SCL superclass** trains SCL on Cifar-100 with 20 superclasses. 2) **Hierarchical Triplet** (*Ge*, 2018), which uses the superclasses to mine hard examples. 3) **Fast AP** (*Cakir et al.*, 2019), a "learning to rank" approach that directly optimizes Average Precision. 4) **Label smoothing** (*Szegedy et al.*, 2016), which reduces network over-confidence and can improve OOD detection (*Lee and Cheon*, 2020). Here, we assign some probability mass to the classes from the same superclass. 5) A multi-classification baseline, referred to as **two heads**, that jointly predicts

Method	Cifar100 fine		Cifar100 superclass
	Accuracy	R@1	R@1
Cross-entropy*	74.52 ± 0.32	74.84 ± 0.21	83.99 ± 0.21
Cross-entropy s.a.*	75.46 ± 1.09	76.03 ± 1.04	84.68 ± 0.86
Triplet	68.44 ± 0.18	47.73 ± 0.14	72.29 ± 0.27
Hierarchical Triplet*	69.27 ± 1.64	65.31 ± 2.69	77.41 ± 1.55
Fast AP*	66.96 ± 0.88	62.03 ± 0.51	69.56 ± 0.54
Smooth Labels	75.66 ± 0.27	74.90 ± 0.06	85.59 ± 0.12
Two heads	74.08 ± 0.40	73.62 ± 0.31	81.92 ± 0.21
SCL-in superclass*	74.41 ± 0.15	69.83 ± 0.28	85.35 ± 0.51
SCL-in*	76.86 ± 0.18	73.20 ± 0.19	82.16 ± 0.24
SCL-out*	76.70 ± 0.29	74.45 ± 0.39	82.94 ± 0.39
SCL-in two heads*	77.15 ± 0.14	74.36 ± 0.10	83.31 ± 0.09
SCL-out two heads*	76.91 ± 0.08	74.87 ± 0.37	83.74 ± 0.16
RINCE-out	76.94 ± 0.16	76.68 ± 0.09	86.10 ± 0.25
RINCE-out-in	$\mathbf{77.59 \pm 0.21}$	$\underline{77.47 \pm 0.16}$	$\underline{86.20 \pm 0.23}$
RINCE-in	$\underline{77.45 \pm 0.05}$	$\mathbf{77.56 \pm 0.03}$	$\mathbf{86.46 \pm 0.21}$

Table 5.2: **Classification and Retrieval Results for Cifar-100 Pretraining.** We perform a classification and retrieval evaluation for a fine-grained task (fine) with 100 classes and a coarse-grained superclass task (superclass) with 20 classes. We report the mean and standard deviation over 3 runs; best method in bold, second best underlined. * indicates methods of others trained by us. All methods use a ResNet-50.



(a) SCL-in (b) RINCE-in (c) RINCE-out-in

Figure 5.4: **Qualitative Comparison of Embedding Spaces.** T-SNE plot of (a) supervised contrastive learning (*SCL-in*) and (b) *RINCE-in* (c) *RINCE-out-in* on Cifar-100. Best seen in color, on screen and zoomed in. The color and marker type combined indicate the class. Labels are omitted for simplicity. RINCE learns a more structured embedding space than SCL, *e.g.* classes are linearly separable and can be modelled well by a Gaussian.

class and superclass labels using two different prediction heads on top of the ResNet-50 backbone. 6) **SCL two heads**, a variant of two heads, that uses the SCL loss instead of cross-entropy.

Classification and Retrieval on Cifar. For the classification evaluation we train a linear layer on top of the last layer of the frozen pre-trained networks. The non-parametric retrieval evaluation involves finding the relevant data points in the feature space of the pre-trained network in terms of class labels via a simple similarity metric, *e.g.* cosine similarity (Eq. (3.4)). RINCE is superior to the baselines for all experiments, see Table 5.2. Note, that all evaluations in Table 5.2 are based on the same pre-trained weights using Cifar-100 fine labels as the first rank and, if applicable, superclass labels as the second rank.

These experiments indicate that training with RINCE maintains ranking order and results in a more structured feature space in which samples of the same class are well separated from other classes. This is further approved by a qualitative comparison between embedding spaces in Figure 5.4.

Furthermore, we find that the grouping of classes is learned by the MLP head h. The increased difficulty of the ranking task of RINCE results in a more structured embedding space before the MLP compared with SCL, see Figure 5.7.

5.3.3 Out-of-Distribution Detection

To further investigate the structure of the learned representation of RINCE, we evaluate on the task of out-of-distribution detection (OOD). As argued in (*Winkens et al.*, 2020), models trained with cross-entropy only need to distinguish classes and can omit irrelevant features. Contrastive learning differs, by forcing the network to distinguish between each pair of samples, resulting in a more complete representation. Such a representation is beneficial for OOD detection (*Hendrycks et al.*, 2019; *Winkens et al.*, 2020). Therefore, OOD performance can be seen as evaluation of representation

Method	AUROC	
	$\mathcal{D}_{\text{out}}$: Cifar-10	$\mathcal{D}_{\text{out}}$: TinyImageNet
Soft Labels$^{\circ}$ (*Lee and Cheon*, 2020)	N/A	67.50
ODIN† (*Liang et al.*, 2018)	77.20	85.20
Mahalanobis† (*Lee et al.*, 2018)	77.50	97.40
Contrastive OOD‡ (*Winkens et al.*, 2020)	78.30	N/A
Gram Matrices (*Sastry and Oore*, 2020)	67.90	98.90
Cross-entropy*	75.32 ± 0.65	77.76 ± 0.77
Cross-entropy s.a.*	75.91 ± 0.10	79.44 ± 0.50
Triplet	70.33 ± 0.54	80.76 ± 0.24
Hierarchical Triplet*	71.97 ± 2.48	76.22 ± 1.27
Fast AP*	69.14 ± 1.02	72.44 ± 0.94
Smooth Labels	74.35 ± 0.65	80.10 ± 0.77
Two heads	$\mathbf{77.99 \pm 0.07}$	78.35 ± 0.39
SCL-in superclass*	74.40 ± 0.72	80.20 ± 1.05
SCL-in*	74.63 ± 0.16	78.96 ± 0.45
SCL-out*	75.32 ± 0.59	79.80 ± 0.70
SCL-in two heads*	75.41 ± 0.16	79.34 ± 0.19
SCL-out two heads*	75.27 ± 0.34	79.64 ± 0.53
Contrastive OOD	74.20 ± 0.40	N/A
RINCE-out	$\underline{77.76 \pm 0.09}$	81.02 ± 0.14
RINCE-out-in	76.82 ± 0.44	$\underline{81.40 \pm 0.38}$
RINCE-in	77.03 ± 0.53	$\mathbf{81.78 \pm 0.05}$

Table 5.3: **OOD Results for Cifar-100 Pretraining.** OOD task with inlier dataset $\mathcal{D}_{\text{in}}$: Cifar-100 and outlier dataset $\mathcal{D}_{\text{out}}$: Cifar-10 and TinyImageNet. We report the mean and standard deviation over 3 runs; best method in bold, second best underlined. Note that, models indicated with † are not directly comparable, since they use data explicitly labeled as OOD samples for tuning. * indicates methods of others trained by us, $^{\circ}$ uses $2\times$ wider ResNet-40, ‡ $4\times$ wider ResNet-50. The lower part of the table uses ResNet-50.

quality beyond standard metrics like accuracy and retrieval. RINCE incentivizes the network to learn an even richer representation.

We follow common evaluation settings for OOD (*Lee et al.*, 2018; *Liang et al.*, 2018; *Winkens et al.*, 2020). Here, Cifar-100 is used as the inlier dataset $\mathcal{D}_{\text{in}}$, Cifar-10 and TinyImageNet as outlier dataset $\mathcal{D}_{\text{out}}$. Note that Cifar-100 and Cifar-10 have disjoint labels and images. For both protocols we only use the test or validation images. Our models are identical to those in the previous section. Inspired by *Winkens et al.* (2020), we follow a simple approach, and fit C n-dimensional class-conditional multivariate Gaussians to the embedding of the training set with (*Pedregosa et al.*, 2011), where n is the dimension of the embedding space and C denotes the number of classes in $\mathcal{D}_{\text{in}}$. We use the log-likelihood to define the OOD-score:

$$s(x) = \max_c(\log(L_c(x))), \tag{5.16}$$

where L_c denotes the likelihood function of the Gaussian for class c. Note that this approach does not require any data explicitly labelled as OOD samples and can be applied out-of-the-box. As a result, the likelihood to identify OOD-samples is high, if each in-class follows roughly a Gaussian distribution in the embedding space, compare Figure 5.4a and 5.4c. We use this approach for all our baselines, *i.e.* cross-entropy, label smoothing, two-heads, SCL-in superclass, SCL-in and SCL-out.

For evaluation, we compute the area under the receiver operating characteristic curve (AUROC). Note that this metric is independent of any threshold and can be directly used on the OOD-scores. An intuitive interpretation of this metric is as the probability that a randomly picked in-distribution sample gets a higher *in-distribution score* than an OOD sample.

Results and a comparison to the most related previous work is shown in Table 5.3. Note that we aim here to compare the learned representation space via RINCE to its counterparts, *i.e.* cross-entropy and SCL, but show well known methods as reference. Most importantly, RINCE clearly outperforms cross-entropy, all SCL variants, contrastive OOD and our own baselines using the identical OOD approach. Only two-heads outperforms all other methods in the near OOD setting with $\mathcal{D}_{\text{out}}$: Cifar-10. However, performance on all other settings is low, showing weak generalization. This underlines our hypothesis, that training with RINCE yields a more structured and general representation space. Comparing to related works, RINCE not only outperforms Contrastive OOD (*Winkens et al.*, 2020) using the same architecture, but even approaches the $4\times$ wider ResNet on Cifar-10 as $\mathcal{D}_{\text{out}}$. ODIN (*Liang et al.*, 2018) and Mahalanobis (*Lee et al.*, 2018) require samples labelled as OOD to tune parameters of the OOD approach. Here we evaluate in the more realistic setting without labelled OOD samples. Despite using significantly less information, RINCE is compatible with them and even outperforms them for $\mathcal{D}_{\text{out}}$: Cifar-10.

5.3.4 Large Scale Data and Noisy Similarities

Additionally, we perform the same evaluations on ImageNet-100, a 100-class subset of ImageNet, see Table 5.4. Here, we use ResNet-18. We obtain the second rank classes for a given class via similarities of the RoBERTa (*Liu et al.*, 2019) class name embeddings. In contrast to the previous experiments, where ground truth hierarchies are known, these similarity scores are noisy and inaccurate – yet it still provides valuable information to the model.

More specifically, to obtain ranking for ImageNet-100 we make use of recent progress in natural language processing. We use the RoBERTa (*Liu et al.*, 2019) implementation provided by *Reimers*

Method	Accuracy	AUROC $\mathcal{D}_{\text{out}}$:ImageNet-100†	AUROC $\mathcal{D}_{\text{out}}$:AwA2
Cross-entropy s.a.	83.94	79.076 ± 1.477	79.04
SCL-out	84.18	79.779 ± 1.274	79.05
RINCE-out-in	**84.90**	$\mathbf{80.473 \pm 1.210}$	**80.73**

Table 5.4: **ImageNet-100 classification accuracy and OOD detection for $\mathcal{D}_{\text{in}}$: ImageNet-100, and $\mathcal{D}_{\text{out}}$: ImageNet-100† and AwA2 (*Xian et al.*, 2018).** ImageNet-100† denotes three ImageNet-100 datasets with non-overlapping classes.

Second Rank Threshold	Accuracy	AUROC
0.35	**59.78**	60.92
0.40	59.40	61.80
0.45	59.44	**62.92**
0.50	59.01	62.05
0.55	58.27	60.54

Table 5.5: **Ablation Study on the RoBERTa Word Similarity Threshold on TinyImageNet.** We use different thresholds to define the second rank positives. We use RINCE-out-in for pretraining and report accuracy of linear evaluation and AUROC for OOD.

and Gurevych (2019)[2] trained for the *semantic textual similarity* benchmark (STSb) (*Cer et al.*, 2017). We use this model to embed each class name into a 1024 dimensional embedding. Class similarities are computed for each pair using the cosine similarity in Eq. (3.4). A small ablation of the robustness to the similarity score threshold is shown in Table 5.5. The accuracy is relatively robust on the choice of this threshold. We find 0.45 to give the best results and use it for the main experiments in Table 5.4.

We evaluate our model via linear classification on ImageNet-100 and two OOD tasks: AwA2 (*Xian et al.*, 2018) as $\mathcal{D}_{\text{out}}$ and ImageNet-100†, where we use the remaining ImageNet classes to define three non-overlapping splits and report the average OOD. Result are shown in Table 5.4. Again, RINCE significantly improves over SCL and cross-entropy in linear evaluation as well as on the OOD tasks. This demonstrates 1) that RINCE can handle noisy rankings and 2) that RINCE leads to improvements on large scale datasets. Next, we move to an even less controlled setting and define

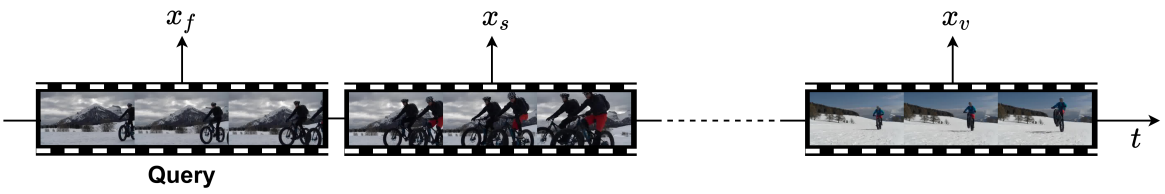


Figure 5.5: **Positives in Videos.** For a given query clip we use *frame* positives x_f, *shot* positives x_s and *video* positives x_v.

Method	Loss	Positives	Negatives	Top 1 Accuracy		Retrieval mAP	
				HMDB51	UCF101	HMDB51	UCF101
InfoNCE	$\mathcal{L}$	$\{x_f\}$	$\mathcal{N}$	41.5	71.3	0.050	0.069
Hard Positive	$\mathcal{L}^{\text{in}}$	$\{x_f, x_s, x_v\}$	$\mathcal{N}$	42.6	74.3	0.069	0.112
	$\mathcal{L}^{\text{out}}$	$\{x_f, x_s, x_v\}$	$\mathcal{N}$	41.4	73.6	0.067	0.120
Easy Positive	$\mathcal{L}^{\text{in}}$	$\{x_f, x_s\}$	$\mathcal{N}$	42.7	74.5	0.058	0.126
	$\mathcal{L}^{\text{out}}$	$\{x_f, x_s\}$	$\mathcal{N}$	40.7	73.5	0.059	0.130
Hard Negative	$\mathcal{L}^{\text{in}}$	$\{x_f, x_s\}$	$\{x_v\} \cup \mathcal{N}$	43.6	74.3	0.068	0.114
	$\mathcal{L}^{\text{out}}$	$\{x_f, x_s\}$	$\{x_v\} \cup \mathcal{N}$	43.5	75.2	0.068	0.119
RINCE	$\mathcal{L}_{\text{RINCE-uni}}$	$x_f > x_s > x_v$	$\mathcal{N}$	**44.9**	**75.4**	**0.072**	**0.140**

Table 5.6: **Finetuning on UCF101 and HMDB51.** $\mathcal{L}$, $\mathcal{L}^{\text{in}}$ and $\mathcal{L}^{\text{out}}$ correspond to Eq. (5.1), Eq. (5.3) and Eq. (5.2), respectively. *Positives* and *Negatives* indicates how x_f, x_s, x_v were incorporated into contrastive learning, where $\mathcal{N}$ denotes the set of negative pairs from random clips. Since we consider only a single positive per rank we use the RINCE-uni loss variant for RINCE.

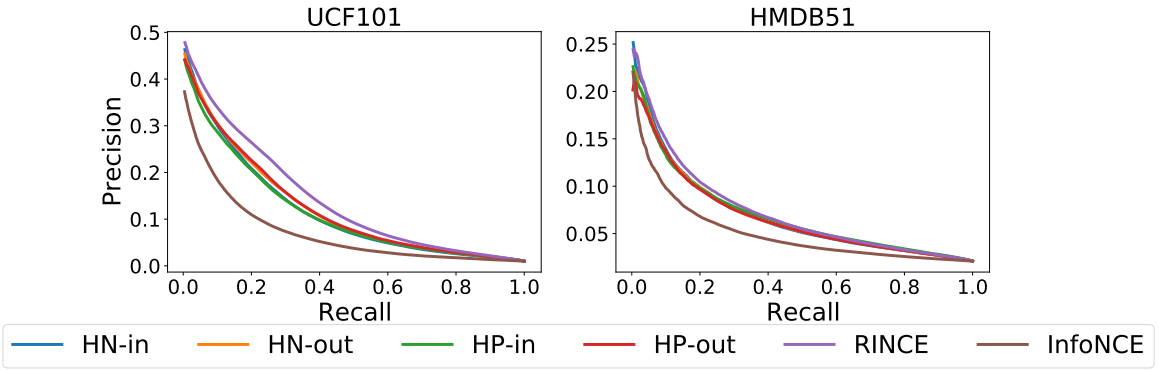


Figure 5.6: **Precision-Recall Curves on UCF101 and HMDB51.** We show the precision-recall curves for the InfoNCE baseline, hard negatives (HN), hard positives (HP) and RINCE on the two action recognition datasets UCF101 and HMDB51. We observe that both the in- and out-option of HN and HP improve over InfoNCE, while RINCE performs best.

a ranking based on temporal ordering for unsupervised video representation learning.

5.3.5 Unsupervised RINCE for Video Representation Learning

In this section we demonstrate that RINCE can be used in a fully unsupervised setting with noisy hierarchies by applying it to an unsupervised video representation task. Inspired by (*Tschannen et al.*, 2020a), we construct three ranks for a given query video: same frames, same shot and same video, see Figure 5.5.

The first positive x_f is obtained by augmenting the query frames. The second positive x_s is a clip consecutive to the query frames, where small transformations of the objects, illumination changes, etc. occur. The third positive x_v is sampled from a different time interval of the same video, which may show visually distinct but semantically related scenes. Naturally, x_f shows the most similar

[2]In particular we use the *stsb-roberta-large* model.

Self-Supervised Methods			top1 Accuracy	
Method	Architecture	Pretraining Dataset	UCF101	HMDB51
BFP, Chapter 4	2D3D-R18	Kinetics-400	66.4	45.3
3DRot (*Jing and Tian*, 2018)	3D-R18	Kinetics-600	62.9	33.7
3D-ST-Puzzle (*Kim et al.*, 2019)	3D-R18	Kinetics-400	65.8	33.7
VIE (*Zhuang et al.*, 2020)	3D-R18	Kinetics-400	72.3	44.8
LA-IDT (*Tokmakov et al.*, 2020)	3D-R18	Kinetics-400	72.8	44.0
RSPNet (*Chen et al.*, 2021a)	3D-R18	Kinetics-400	74.3	41.8
Temp Trans (*Jenni et al.*, 2020)	3D-R18	Kinetics-600	79.3	49.8
MFO† (*Qian et al.*, 2021a)	3D-R18	Kinetics-400	79.1	47.6
TimeEq† (*Jenni and Jin*, 2021)	3D-R18	Kinetics-400	**87.1**	**63.6**
ASCNet† (*Huang et al.*, 2021)	3D-R18	Kinetics-400	80.5	52.3
TCLR† (*Dave et al.*, 2022)	3D-R18	Kinetics-400	85.4	55.4
RINCE	3D-R18	Kinetics-400	75.4	44.9

Table 5.7: **Finetuning Evaluation on UCF101 and HMDB51.** We compare our RINCE method to other self-supervised video representation learning methods via finetuning on UCF101 and HMDB51 split 1. † denotes methods that were published concurrently or subsequently to RINCE.

content to the query frames, followed by x_s and finally x_v. We compare temporal ranking with RINCE to different baselines.

Baselines. We compare to the basic **InfoNCE**, where a single positive is generated via augmentations (*Chen et al.*, 2020b; *He et al.*, 2020), *i.e.* only *frame* positives x_f. When considering multiple clips from the same video such as x_s and x_v, there are several possibilities: We can treat them all as positives (**hard positive**), we can use the distant x_v as a **hard negative** or ignore it (**easy positive**). In both cases $\mathcal{L}^{\text{out}}$, Eq. (5.2), and $\mathcal{L}^{\text{in}}$, Eq. (5.3), are possible. Additionally, we compare our method to other self-supervised video representation learning methods that are trained in comparable settings using similar network architecture and pretraining dataset, including our BFP method from Chapter 4.

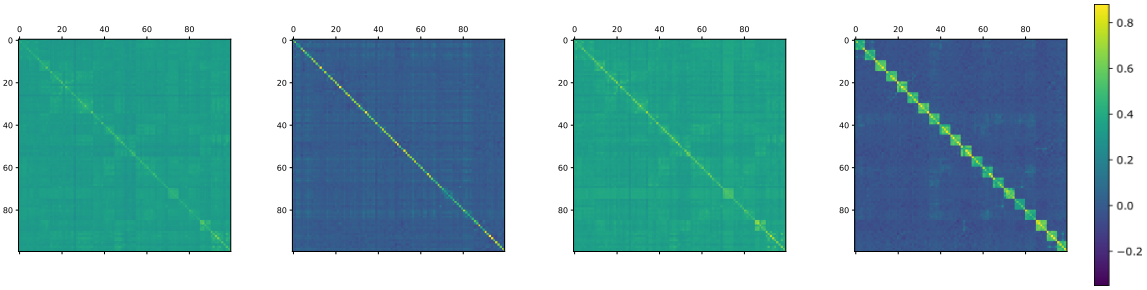
Ranking Frame-, Shot- and Video-level Positives. We sample short clips of a video, each consisting of 16 frames sampled with a temporal stride of 3 for x_f, x_s and x_v. We ensure a gap of at least 48 frames between x_v and the other two positives x_f and x_s. We augment each clip with a set of standard video augmentations: random sized crop of size 128×128, horizontal flip, color jittering and random color drop. For the anchor clip x, we define positives as in Figure 5.5: $p_1 = x_f$ consists of the same frames as x, $p_2 = x_s$ is a sequence of 16 frames adjacent to x, and $p_3 = x_v$ is sampled from a different time interval than x_f and x_s. Negatives x_n are sampled from different videos. Since each rank i contains only a single positive p_i, the different InfoNCE variants Eq. (5.1), Eq. (5.2) and Eq. (5.3) are the same, we call this specific setting RINCE-uni. By ranking the positives we ensure that the similarities satisfy $\text{sim}(x, x_f) > \text{sim}(x, x_s) > \text{sim}(x, x_v) > \text{sim}(x, x_n)$, adhering to the temporal structure in videos.

Implementation Details. We use a 3D-Resnet18 backbone (*Hara et al.*, 2018) in all experiments and pool the feature map into a single 512-dimensional feature vector. The MLP head h has 512 hidden units with ReLU activation. Note that the MLP head is removed after self-supervised training and will not be transferred to downstream tasks. We use the Adam optimizer (*Kingma and Ba*, 2015) with weight decay $1e{-5}$, a batch size of 128 and an initial learning rate of $1e{-3}$, that is decreased by a factor of 10 when the validation loss plateaus, and train for 200 epochs. We use a memory bank size of 65.536 (*He et al.*, 2020) and do the momentum update with $m = 0.99$. We use a temperature parameter $\tau = 0.1$ for the baselines (InfoNCE, hard positives, hard negatives) and $\tau_1 = 0.1, \tau_2 = 0.15, \tau_3 = 0.2$ for RINCE.

Datasets and Evaluation. For self-supervised learning, we use Kinetics-400 (*Kay et al.*, 2017) and discard the labels. Our version of the dataset consists of 234.584 training videos. We evaluate the learned representation via finetuning on split 1 of UCF101 (*Soomro et al.*, 2012) and HMDB51 (*Kuehne et al.*, 2011) and report top1 accuracy. In this evaluation, the pretrained weights are used to initialize the backbone with an added randomly initialized linear layer and train it end-to-end using cross-entropy. We finetune the models for 500 epochs using the Adam optimizer with weight decay $1e{-5}$ and a learning rate of $1e{-4}$ that is reduced by a factor of 10 when the validation loss plateaus. Furthermore, we use a dropout rate of 0.9 on the output of the linear layer. Additionally, we evaluate the representation via nearest neighbor retrieval and report mAP; precision-recall curves can be found in Figure 5.6.

Experimental Results. We report the results for RINCE as well as the baselines in Table 5.6. Adding shot- and video-level samples to InfoNCE improves the downstream accuracies and retrieval. We observe that adding x_v to the set of negatives to provide a hard negative rather than adding it to the set of positives leads to higher performance, suggesting that this should not be a true positive. This is further supported by the second and third row, where all three positives are treated as true positives. Here, $\mathcal{L}^{\text{out}}$, which forces all positives to be similar, leads to inferior performance compared to $\mathcal{L}^{\text{in}}$. $\mathcal{L}^{\text{in}}$ allows more noise in the set of positives by weak influence of false positives x_v. With RINCE we can impose the temporal ordering $x_f > x_s > x_v$ and treat x_v properly, leading to the highest downstream performance. Improvements of RINCE over $\mathcal{L}^{\text{out}}$ are less pronounced on UCF101. We conjecture that this is due to the strong static bias (*Li et al.*, 2018) of UCF101 and $\mathcal{L}^{\text{out}}$ encourages static features. Contrarily, improvements of RINCE over $\mathcal{L}^{\text{out}}$ on HMDB51 are substantial, due to the weaker bias towards static features. The precision-recall curves in Figure 5.6 further validate these findings. We observe that the hard positive (HP) and hard negative (HN) baselines improve over InfoNCE, and RINCE outperforms all baselines.

Finally, we compare our method to other unsupervised video representation learning methods that use a similar backbone network in Table 5.7. While we outperform several competitive methods at the time of publication, *e.g.* VIE (*Zhuang et al.*, 2020) and LA-IDT (*Tokmakov et al.*, 2020), more recent works have since been published that outperform RINCE by large margins. This shows that, although RINCE improves over the vanilla InfoNCE loss, different approaches to incorporate the temporal information into contrastive video representation learning is promising. For example, *Jenni and Jin* (2021) incorporate the temporal structure of videos in several different ways: They encode the *relative* temporal transformation between two clips and use them as a positive to the same relative temporal transformation of a different video. This puts an explicit focus on the temporal



(a) SCL-in before h (b) SCL-in after h (c) RINCE-in before h (d) RINCE-in after h

Figure 5.7: **Similarity Matrix on Cifar-100 Classes Before and After the MLP Head h.** Classes are sorted, such that superclasses are grouped together. Similarity values are the average cosine similarity between classes. Similarity is larger for RINCE after the MLP, suggesting that the ranking is learned by the MLP.

transformation rather than the video content alone, but does not work well on its own. Instead, they combine the contrastive loss with more traditional self-supervised learning pretext tasks that predict the temporal transformation that has been applied to the video. In combination, they achieve very strong performance, which demonstrates the effectiveness of combining the best of both worlds, *i.e.* contrastive learning and temporal pretext tasks.

Notably, RINCE significantly improves over our BFP method from Chapter 4 on the UCF101 dataset, while achieving comparable performance on the HMDB51 dataset. BFP is explicitly designed to capture temporal features and less focused on static features, while RINCE does not impose any explicit constraints to learn temporal features and may end up improving mostly on the static video features, which are more beneficial for the UCF101 dataset (*Li et al.*, 2018). Note that a direct comparison between BFP and RINCE is not possible as the methods were trained using different backbone networks, namely a 2D3D-Resnet18 for BFP and a 3D-Resnet18 for RINCE. A fairer comparison would require to train the BFP method with a 3D-Resnet18 backbone. As demonstrated in Section 4.3.4, the BFP method relies on and is sensitive to the set of temporal hard negatives; we found that an ideal setting consists of 2 past and future clips, respectively, and 2 to 4 present clips, where each clip consists of 5 frames. However, the 3D-Resnet18 architecture used in this chapter takes input clips of 16 frames, which we find to be computationally infeasible with the number of clips required for training BFP.

5.3.6 Does RINCE Rank Samples?

The experiments above indicate that RINCE leads to a well structured embedding space with useful properties, but whether RINCE learns to rank was evaluated only indirectly. To empirically validate that our method preserves the desired ranking we compute the average cosine distance, Eq (3.4), between classes. If ranking is encoded in the embedding, the highest similarity should be seen for same class, the second highest for samples from the same superclass and a low similarity for others. We visualize the results as similarity matrix in Figure 5.7. It can be seen that RINCE-in learns the ranking to some extent in the embedding space (Figure 5.7c). The differences become more apparent after the MLP head h. Here, superclasses show high similarity, which however, does not approach the within-class-similarity (compare Figure 5.7c and 5.7d). This shows that ranking is implemented

to a large extent in the MLP. SCL-in also tends to have slightly higher similarity within superclasses (Figure 5.7a), but after the MLP mostly within-class-similarity is preserved (Figure 5.7b). This observation confirms that Eq. (5.8) mirrors the desired ranking in the latent space. Enforcing such a structure in the output space, results in a better representation in the intermediate layer, as the previous layer should explore the underlying structure of data more intensively to provide the last layers of low capacity with enough information for the ranking.

5.4 Conclusion

We introduced RINCE, a new member in the family of InfoNCE losses. We show that RINCE can exploit rankings to learn a more structured feature space with desired properties, lacking with standard InfoNCE. Furthermore, representations learned through RINCE can improve accuracy, retrieval and Out-of-Distribution detection. Most importantly, we show that RINCE works well with noisy similarities, is applicable to large scale datasets and to unsupervised training. We compare the different variants of RINCE. Here lies a limitation: Different variants are optimal for different tasks and must be chosen based on domain knowledge. Future work will explore further applications of obtaining similarity scores, *e.g.* based on distance in a pretrained embedding space, distance between cameras in a multi-view setting or distances between clusters.

Both the BFP method from Chapter 4 and RINCE aim to learn a single video representation and demonstrate favorable performance on different datasets: BFP is explicitly designed to focus on temporal features in videos and performs well on the HMDB51 dataset. On the other hand, RINCE aims to capture features that change gradually over time, which may accommodate static features better, and shows a stronger performance on the UCF101 dataset. In contrast to these methods that focus on a single task or only a specific type of video features, we aim to learn different types of features in a single representation in Chapter 6.

CHAPTER 6

Long Short View Feature Decomposition via Contrastive Video Representation Learning

In the previous two chapters, we have introduced two contrastive learning methods for learning high quality video representations. While Chapter 4 is focused on learning temporally structured representations via past and future feature prediction, Chapter 5 introduces a general contrastive learning method that allows us to incorporate a ranked set of positives, ranking temporally closer video clips higher than temporally far clips. Still, both methods aim to learn a single video representation, which may be sub-optimal when addressing a diverse set of downstream tasks. In particular, our BFP method in Chapter 4 aims to learn temporally structured representations that capture temporal variations of the data well. Meanwhile, practical applications may require both temporally varying as well as non-varying, temporally consistent features, which we call stationary and non-stationary features, respectively. In this chapter, we propose a method to learn both types of features. Learning such diverse features in a single representation may be sub-optimal; therefore, we propose to decompose the representation space into stationary and non-stationary features. To that end, we construct positive pairs from long and short views (*i.e.* longer video sequences and their shorter sub-sequences) that encourage the model to learn stationary and non-stationary features in videos. Namely, stationary features remain the same over time and are shared between long and short view, constituting our first positive pair; non-stationary features on the other hand vary – the long view non-stationary features are paired with an aggregated form of the short view non-stationary features. We evaluate the proposed method extensively on several downstream tasks and analyse the learned representations. We find that the different types of features are beneficial for different types of downstream tasks: Stationary features, which remain similar throughout the video, enable the prediction of video-level action classes. Non-stationary features, which represent temporally varying attributes, are more beneficial for downstream tasks involving more fine-grained temporal understanding, such as action segmentation.

Individual Contribution

The following chapter is based on the publication (*Behrmann et al.*, 2021a):

Long Short View Feature Decomposition via Contrastive Video Representation Learning
Nadine Behrmann, Mohsen Fayyaz, Juergen Gall, and Mehdi Noroozi
IEEE International Conference on Computer Vision (ICCV), 2021.

This publication was done in close collaboration between Mohsen Fayyaz, Mehdi Noroozi and Nadine Behrmann. Juergen Gall provided scientific guidance and very valuable feedback and suggestions. Mehdi Noroozi proposed the underlying idea for the non-stationary contrastive loss. In discussions between Mehdi Noroozi and Nadine Behrmann, the idea was further extended to include stationary features as well. The feature decomposition approach was proposed by Mehdi Noroozi.

The implementation of the self-supervised learning stage was largely done by Nadine Behrmann and occasionally supported by Mohsen Fayyaz. The standard evaluations via action recognition downstream tasks and video retrieval were implemented and carried out by Nadine Behrmann. The evaluation via temporal action segmentation was proposed in joint discussions between Mehdi Noroozi, Mohsen Fayyaz, Juergen Gall and Nadine Behrmann and implemented by Mohsen Fayyaz. An analysis besides evaluations on downstream tasks was suggested by a reviewer; the specific analyses carried out were designed by Mehdi Noroozi and Nadine Behrmann and implemented by Nadine Behrmann.

Contents

6.1 Introduction

Learning rich video representations is a key challenge for general video understanding. An ideal representation extracts useful information that benefits numerous downstream tasks such as action recognition, retrieval and action segmentation. Learning such representations in a supervised setting is inherently biased towards static features (*Li et al.*, 2018). However, in order to solve more complex downstream tasks, that require temporal attributes of videos such as temporal action segmentation, we need a more diverse set of features. As a remedy, we train our network to represent stationary and non-stationary features. To get an intuition, let's consider the following example: a video of a bartender mixing a cocktail in Figure 6.1 is stationary in one sense – we see a bartender in a bar with liquor bottles in the background – but non-stationary in another: different steps in preparing a cocktail are shown, such as holding the bottle, pouring the left bottle and pouring the right bottle. Here, the stationary attributes of the video enable us to predict the overall action class, *i.e.* mixing

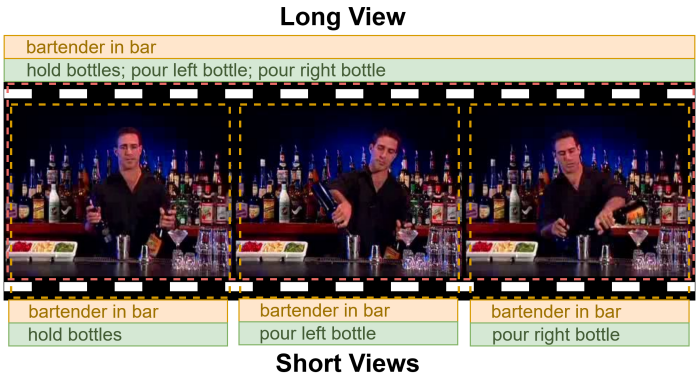


Figure 6.1: Video attributes can be divided into two sections: Stationary features (shown in yellow), that are shared between long and short views, and non-stationary features (shown in green), that aggregate the short views to match the corresponding long view.

cocktails. On the other hand, the non-stationary attributes enable more fine-grained temporal distinctions, *e.g.* predicting when different steps occur in the video. Ideally, both types of attributes should be represented by video models.

Learning representations in a supervised setting, usually involves pretraining on large-scale labeled datasets such as Kinetics (*Kay et al.*, 2017) with video-level annotations, inhibiting strong static biases (*Li et al.*, 2018). As a result they capture mainly the stationary features, but largely ignore non-stationary features as the stationary features are sufficient for action recognition. Self-supervised leaning provides a promising direction to address this shortcoming. As the supervisory signals arise from the underlying structure of the data, it has the potential to extract more descriptive features. Several previous self-supervised methods aim to capture temporal features in videos by designing a temporal pretext task, *e.g.* predicting temporal transformations (*Jenni et al.*, 2020) or video speed (*Benaim et al.*, 2020). These methods are not explicitly encouraged to capture stationary and non-stationary features. In contrast, we explicitly decompose the representation space into stationary and non-stationary features, enabling us to solve a more diverse set of downstream tasks, including action recognition and temporal action segmentation.

Following the recent trend in self-supervised learning, our proposed method fits in the contrastive learning framework. The supervisory signal that leads us to distinguish between the stationary and non-stationary features emerges from long and short views of a given video. Naively applying contrastive learning to long and short views results in a set of features that represent both long and short views similarly. We argue that this assumption is only valid for a subset of features, which we call stationary. The other subset, which we call non-stationary features, includes a set of features that aggregate from short to long views, *i.e.* combining attributes of several short views gives us the attributes of the long view, see Figure 6.1. Therefore, imposing a naive similarity between the long and short views is prone to ignoring the non-stationary features, which are crucial for different downstream tasks. Accordingly, we divide the final features into two disjoint subsets, which are later used in two separate contrastive losses. For the stationary features of a given long view, we provide a positive pair through the stationary features of a corresponding short view. Whereas an aggregated form of the non-stationary features of all corresponding short views forms a positive pair for the

non-stationary features of the long view.

We validate our argument above empirically and train our method on the Kinetics dataset (*Kay et al.*, 2017) without using any labels. We evaluate the model on several downstream tasks and datasets and analyse the learned representations; we highlight the main results here. Our proposed method achieves state-of-the-art retrieval results on the HMDB51 dataset, and outperforms a contrastive learning baseline using data augmentation by significant margins on the UCF101 (*Soomro et al.*, 2012) and HMDB51 (*Kuehne et al.*, 2011) datasets for action classification: 78.0% and 53.7% accuracy versus 72.7% and 46.3%, respectively. Additionally, we evaluate the learned representations for temporal action segmentation on the Breakfast dataset (*Kuehne et al.*, 2014). We show that, our non-stationary features perform substantially better than the stationary features on this dataset, supporting our hypothesis on the design choice. To our knowledge, we are the first to conduct this evaluation for video representation learning. Furthermore, we analyse the learned representations and find that stationary features capture attributes, that remain similar over time and can be detected in a few frames, while non-stationary features encompass more temporally varying attributes, which are revealed when observing more frames.

Our contributions are as follows: 1) We propose the novel approach of **L**ong **S**hort View **F**eature **D**ecomposition (LSFD). 2) Our proposed method achieves state-of-the-art retrieval performance on HMDB51 using a 3D-Resnet18. 3) We evaluate our proposed method on a temporal action segmentation task for the first time, and show that it outperforms the supervised baseline with high margins across multiple metrics. 4) We investigate the feature decomposition capability of our method and analyse the learned representations. We find that stationary features remain more stable over time and perform better on the tasks and datasets that include large static biases, *e.g.* action classification on UCF101. On the other hand, non-stationary features vary over time and are beneficial for more dynamic tasks and datasets, *e.g.* temporal action segmentation on the Breakfast dataset.

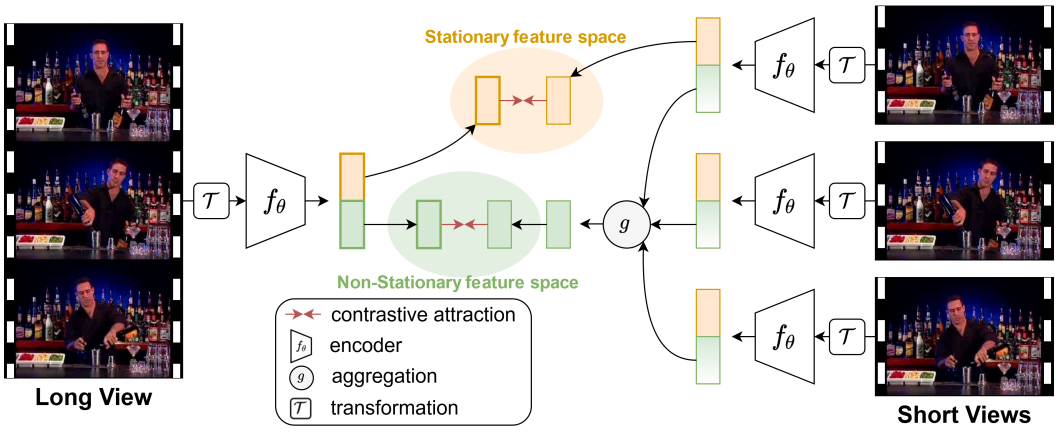


Figure 6.2: We extract features from long and short views and decompose them into stationary and non-stationary features, shown in orange and green, respectively. Stationary features remain similar over time and are shared by the long and short view, and serve as a positive pair (indicated by red arrows). Non-stationary features on the other hand capture temporal variances; we aggregate non-stationary features of short views to obtain the positive for the long view.

6.2 Method

The key ideas of our **L**ong **S**hort View **F**eature **D**ecomposition (LSFD) method lie in decomposing the video representation into stationary and non-stationary features, and constructing short and long views of the data, see Figure 6.2. Short views provide local attributes, as they span a limited temporal receptive field, while global temporal attributes are better perceived through the larger temporal receptive field of long views. Therefore, a naive solution of imposing similarity between short and long views is not necessarily optimal. We propose to establish a connection between long and short views by decomposing the representation space into two sections: One section represents stationary features that are shared between short and long views. The other section represents the attributes that should aggregate short views to the corresponding long view, which we call non-stationary features. As a result, the network is allowed to establish a proper connection between short and long views, without being forced to represent them similarly. We impose the concept of stationary and non-stationary features via separate contrastive losses on each section of the feature space. In the following, we discuss different components and design choices of our method in more detail.

6.2.1 Stationary and Non-Stationary Features

For a given sequence of video frames, we obtain a *long* view x_l that consists of all frames, and N non-overlapping sub-sequences, $x_s^{(1)}, \dots, x_s^{(N)}$, which serve as *short* views. This allows us to construct positive pairs for stationary and non-stationary features, see Figure 6.2.

More precisely, we train a parametric function f_θ that takes a sequence of videos frames and maps them to a representation space: $f_\theta(x) = \xi = (\psi, \phi)$, where ψ, ϕ denote the stationary and non-stationary features of x, respectively. We compute the features of the long and short views by feeding them to a shared backbone:

$$\begin{aligned} f_\theta(x_l) &= \xi_l = (\psi_l, \phi_l), \\ f_\theta(x_s^{(i)}) &= \xi_s^{(i)} = (\psi_s^{(i)}, \phi_s^{(i)}) \quad \text{for } i \in \{1, \dots, N\}. \end{aligned}$$

This allows us to establish a connection between long and short views. Stationary features represent attributes that remain the same over time, while non-stationary features are aggregated over time. Overall, they satisfy the following two properties:

$$\psi_l \simeq \psi_s^{(i)} \quad \text{for } i \in \{1, \dots, N\}, \tag{6.1}$$

$$\phi_l \simeq g(\phi_s^{(1)}, \dots, \phi_s^{(N)}). \tag{6.2}$$

Here, the aggregation function g can be any function that takes the non-stationary features of the short views and maps them to non-stationary features of the long view. Choosing an appropriate aggregation function is non-trivial and deserves extensive investigation. Candidates range from simple functions, such as a sum, to more elaborate and learnable functions, for example a linear transformation, MLP, or a recurrent network. We provide an ablation of different aggregation functions in Section 6.3.3.

In order to enforce the similarities in Eqs. (6.1) and (6.2) during training, we construct the following positive pairs to be used in two separate contrastive losses. The first pair $(\psi_s^{(j)}, \psi_l)$ aims at features shared between the long and a short view according to Eq. (6.1). The second pair

$(g(\phi_s^{(1)}, \ldots, \phi_s^{(N)}), \phi_l)$ corresponds to Eq. (6.2), which aggregates the non-stationary features of the short views to match the non-stationary features of the long view via an aggregation function g.

For any given pair of features (z_1, z_2) obtained through the described procedure above, we compute the similarity following the recent trend in contrastive learning (*Chen et al.*, 2020b): We apply a learnable transformation h, here an MLP head, and a temperature parameter τ to scale the cosine similarity, denoted by

$$\text{sim}_h(z_1, z_2) = \frac{1}{\tau} \frac{h(z_1)^T h(z_2)}{\|h(z_1)\| \|h(z_2)\|}. \tag{6.3}$$

6.2.2 Training Objective

Our training objective consists of three separate InfoNCE losses applied on stationary, non-stationary, and full features:

$$\mathcal{L} = \mathcal{L}_{\text{stationary}} + \mathcal{L}_{\text{non-stationary}} + \mathcal{L}_{\text{instance}}, \tag{6.4}$$

which we will discuss below. We use three different MLP heas h_s, h_n, and h_i for the three separate losses. We use a set of negatives that consists of random videos:

$$\mathcal{N} = \{f_\theta(\bar{x}_l) = \xi_{\text{neg}} = (\psi_{\text{neg}}, \phi_{\text{neg}}) | \bar{x}_l \text{ is a random video}\}.$$

To avoid shortcuts via low-level video statistics, we apply the same set of standard video augmentations, including random resized crop, horizontal flip, and color augmentation, to long and short views independently.

Stationary Loss. Attributes that remain the same over time – such as non-moving objects or the background scene – are shared between the long view and all short views; and thus should be represented similarly. Therefore, the stationary features of a short view should capture the same attributes as the stationary features of the long view. We encourage such a property by applying the following loss function:

$$\mathcal{L}_{\text{stationary}} = -\log \frac{\exp(\text{sim}_{h_s}(\psi_s^{(j)}, \psi_l))}{\sum_{\bar{\psi}_l \in \mathcal{N}_\psi \cup \{\psi_l\}} \exp(\text{sim}_{h_s}(\psi_s^{(j)}, \bar{\psi}_l))}. \tag{6.5}$$

Here, $\psi_s^{(j)}$ are the stationary features of a randomly selected short view, and $\mathcal{N}_\psi = \{\psi_{\text{neg}} | (\psi_{\text{neg}}, \phi_{\text{neg}}) \in \mathcal{N}\}$.

Non-Stationary Loss. Complementary to stationary features, the non-stationary features represent the content of the video that varies: moving objects and persons, motion, temporal changes in the scene, etc. These temporal changes *aggregate* over time, *i.e.* the non-stationary features of the long view should capture the temporal changes happening in all sub-sequences. We encourage such a property by applying the following loss function:

$$\mathcal{L}_{\text{non-stationary}} = -\log \frac{\exp(\text{sim}_{h_n}(\phi_g, \phi_l))}{\sum_{\bar{\phi}_l \in \mathcal{N}_\phi \cup \{\phi_l\}} \exp(\text{sim}_{h_n}(\phi_g, \bar{\phi}_l))} \tag{6.6}$$

where $\phi_g = g(\phi_s^{(1)}, \ldots, \phi_s^{(N)})$ is an aggregated version of the short view non-stationary features, and $\mathcal{N}_\phi = \{\phi_{\text{neg}} | (\psi_{\text{neg}}, \phi_{\text{neg}}) \in \mathcal{N}\}$.

Instance Recognition Loss. Finally, we add an instance recognition loss on the long views by applying InfoNCE on the full features. To that end we obtain a second view of the long video sequence $\hat{\xi}_l$ via standard video augmentations. This corresponds to the standard contrastive learning approach.

$$\mathcal{L}_{\text{instance}} = -\log \frac{\exp(\text{sim}_{h_i}(\xi_l, \hat{\xi}_l))}{\sum_{\bar{\xi}_l \in \mathcal{N} \cup \{\hat{\xi}_l\}} \exp(\text{sim}_{h_i}(\xi_l, \bar{\xi}_l))}. \tag{6.7}$$

6.3 Experiments

We now evaluate our LSFD method on different downstream tasks. Previous methods evaluate learned representations based on their performance on action recognition tasks; most commonly, models are finetuned on UCF101 and HMDB51. Despite the practical value of finetuning, it is an uncontrolled evaluation (*He et al.*, 2019) and prone to overfitting. Moreover, action classification provides a rather incomplete assessment of the learned representations due to static biases in these datasets (*Li et al.*, 2018). A sparse, global frame sampling strategy, proposed in (*Wang et al.*, 2019b), works well for action recognition; even in the extreme case where only a single frame is used, accuracy remains high on UCF101. This suggests that temporal information is less important for these tasks. To get a better understanding of our representations, we extend the current evaluation protocol by adding another transfer learning task: action segmentation.

In contrast to action recognition, in which a single action label per video is given, action segmentation uses fine-grained temporal annotations. As the scene and background often stay the same throughout the video, a better temporal understanding is needed in order to temporally segment a video into the different actions that occur. Additionally, our pretrained model is frozen in this evaluation and serves as a feature extractor – no finetuning is involved.

To extend the evaluation via downstream tasks, we analyse the properties of the learned representations more thoroughly. Here, we are aiming to get a better understanding of which types of attributes are represented by stationary and non-stationary features, and investigate how and why they are different.

6.3.1 Implementation Details

Architecture. We use a 3D-Resnet18 backbone (*Hara et al.*, 2018) unless otherwise noted and pool the feature map into a single 512-dimensional feature vector. We decompose this feature vector

Self-Supervised Methods			top1 Accuracy	
Method	Architecture	Pretraining Dataset	UCF101	HMDB51
Shuffle&Learn (*Misra et al.*, 2016)	CaffeNet	UCF101	50.2	18.1
OPN (*Lee et al.*, 2017)	VGG	UCF101	59.8	23.8
VCOP (*Xu et al.*, 2019)	R3D	UCF101	64.9	29.5
PRP (*Yao et al.*, 2020)	R3D	UCF101	66.5	29.7
BFP, Chapter 4	2D3D-R18	Kinetics-400	66.4	45.3
DPC (*Han et al.*, 2019)	2D3D-R34	Kinetics-400	75.7	35.7
Pace Pred (*Wang et al.*, 2020b)	R(2+1)D	Kinetics-400	77.1	36.6
MemDPC (*Han et al.*, 2020b)	2D3D-R34	Kinetics-400	78.1	41.2
CBT (*Sun et al.*, 2019)	S3D	Kinetics-600	79.5	44.6
SpeedNet (*Benaim et al.*, 2020)	S3D-G	Kinetics-400	81.1	48.8
CoCLR (*Han et al.*, 2020a)	S3D-G	Kinetics-400	87.9	54.6
MCL† (*Li et al.*, 2021b)	R(2+1)D	Kinetics-400	93.4	69.1
CATE† (*Sun et al.*, 2021)	3D-R50	Kinetics-400	88.4	61.9
CVRL† (*Qian et al.*, 2021b)	3D-R50	Kinetics-400	92.9	67.9
BraVe† (*Recasens et al.*, 2021)	3D-R50	Kinetics-400	93.7	72.0
3D-ST-Puzzle (*Kim et al.*, 2019)	C3D	Kinetics-400	61.2	28.3
MA Stats (*Wang et al.*, 2019a)	C3D	Kinetics-400	61.2	33.4
Temp Trans (*Jenni et al.*, 2020)	C3D	Kinetics-600	69.9	39.6
RSPNet (*Chen et al.*, 2021a)	C3D	Kinetics-400	76.7	44.6
LSFD (Ours)	C3D	Kinetics-400	**79.8**	**52.1**
3DRot (*Jing and Tian*, 2018)	3D-R18	Kinetics-600	62.9	33.7
3D-ST-Puzzle (*Kim et al.*, 2019)	3D-R18	Kinetics-400	65.8	33.7
VIE (*Zhuang et al.*, 2020)	3D-R18	Kinetics-400	72.3	44.8
LA-IDT (*Tokmakov et al.*, 2020)	3D-R18	Kinetics-400	72.8	44.0
RSPNet (*Chen et al.*, 2021a)	3D-R18	Kinetics-400	74.3	41.8
RINCE, Chapter 5	3D-R18	Kinetics-400	75.4	44.9
Temp Trans (*Jenni et al.*, 2020)	3D-R18	Kinetics-600	79.3	49.8
MFO† (*Qian et al.*, 2021a)	3D-R18	Kinetics-400	79.1	47.6
TimeEq† (*Jenni and Jin*, 2021)	3D-R18	Kinetics-400	**87.1**	**63.6**
ASCNet† (*Huang et al.*, 2021)	3D-R18	Kinetics-400	80.5	52.3
TCLR† (*Dave et al.*, 2022)	3D-R18	Kinetics-400	85.4	55.4
LSFD (Ours)	3D-R18	Kinetics-400	77.2	53.7

Table 6.1: **Finetuning on UCF101 and HMDB51.** We compare our LSFD method to previous methods via finetuning on UCF101 and HMDB51 split 1. The first block shows methods with different architectures and pretraining datasets, while the last two blocks encompass methods with the same architecture and pretraining dataset as our method. † denotes methods that were published concurrently or subsequently to LSFD.

into two equal chunks (of size 256) of stationary and non-stationary features. The MLP head h_i has 512 hidden units with ReLU activation, h_s and h_n have 256 hidden units. Note that the MLP heads are removed after self-supervised training and will not be transferred to downstream tasks.

In Section 6.3.3, we investigate the influence of different aggregation functions, namely Sum, Linear, MLP, and GRU. The Sum simply takes the sum over the non-stationary features of the short views, *i.e.* $\sum_{i=1}^{N} \phi_s^{(i)}$. For Linear and MLP we first concatenate the non-stationary features, *i.e.* $\left(\phi_s^{(1)}, \ldots, \phi_s^{(N)}\right) \in \mathbb{R}^{N \times 256}$, and then apply a linear layer or an MLP, respectively, mapping from $\mathbb{R}^{N \times 256}$ to $\mathbb{R}^{256}$. The MLP has $N \times 256$ hidden units with `ReLU` activation. The GRU aggregates the sequence of non-stationary features $\left(\phi_s^{(i)}\right)_{i=1}^{N}$ to produce a single aggregated feature $\phi_g \in \mathbb{R}^{256}$ of the same dimension as the non-stationary features of the long view. We use a one-layer ConvGRU with a kernel size of 1.

Implementation Details. For self-supervised pretraining, we use the Adam optimizer (*Kingma and Ba*, 2015) with weight decay $1e-5$, a batch size of 128 and an initial learning rate of $1e-3$, that is decreased by a factor of 10 when the validation loss plateaus. We set $\tau = 0.1$ and $m = 0.99$ for the momentum update of the key encoder. We use a memory bank size of 65.536 as in (*He et al.*, 2020).

For finetuning we sample clips of 16 frames with a temporal stride of 3, and train the model end-to-end using the standard cross entropy loss. We train for 500 epochs using the Adam optimizer with an initial learning rate of $1e-4$, which we reduce at epoch 300 and 400 by a factor of 10. Weight decay is set to $1e-5$ and we use a dropout of 0.9. During inference we sample 10 clips from each test video, and use ten crop. The predictions are averaged to produce the final prediction of each test video.

Views and Augmentations. We construct long views by sampling $N \cdot L$ frames with a temporal stride of 3. We set $L = 8$ in all experiments, unless otherwise noted and provide experiments with different values of N. Given a long view of $N \cdot L$ frames, we divide them into N non-overlapping sub-sequences of L frames, which serve as short views. We apply spatial augmentations, such as crop and horizontal flip, and color augmentations to each view independently. More specifically, we use random resized crop with probability $p = 1.0$, where a spatial patch is selected covering 50% to 100% of the original frame with an aspect ratio between $3/4$ and $4/3$. Then, we resize the patch to the size 128×128. Horizontal flip is applied with probability $p = 0.5$. For color augmentations we use random color drop with probability $p = 0.1$, and apply color jitter with probability $p = 1.0$, where brightness, contrast, saturation and hue are shifted. We use a maximum brightness adjustment of 0.5, contrast of 0.5, saturation of 0.5, and hue of 0.25. The different views of a video sequence (long and short views) are augmented independently, but within a single view, the frames are augmented consistently, *i.e.* the same crop, color augmentation, etc. is selected for all frames of this view. During finetuning we keep the random crop and horizontal flip, but only apply color jitter as described above with a probability of $p = 0.3$.

Datasets. We conduct experiments on four video datasets. For self-supervised learning, we use videos of **Kinetics-400** (*Kay et al.*, 2017) and discard the labels. Our copy of the dataset consists of 234.584 training and 12.634 validation videos. We evaluate the learned representation on

UCF101 (*Soomro et al.*, 2012) and **HMDB51** (*Kuehne et al.*, 2011) for action recognition and on the **Breakfast** dataset (*Kuehne et al.*, 2014) for action segmentation.

Baseline. Our most important baseline is a contrastive learning baseline, trained with $\mathcal{L}_{\text{instance}}$, Eq. (6.7). This corresponds to traditional contrastive learning (*He et al.*, 2020) applied to videos without an explicit focus on temporal variations. Here, we use $L = 16$ frames and obtain two views by applying augmentations independently. We use the same augmentations mentioned above, including spatial an color augmentations. Specifically, these do not include any temporal augmentations such as temporal crop or varying speed, as it would encourage the representations to be invariant to these augmentations. However, motion patterns across different temporal crops vary and training the model to be invariant to them encourages to discard these features. Similarly, the speed and duration of movements provide valuable information.

Evaluation. To evaluate the learned video representations, we follow the most widely adopted approach of *finetuning*: We use the pretrained weights to initialize the 3D-Resnet18 backbone network, add a randomly initialized linear layer for classification and then train it end-to-end on split 1 of UCF101 and HMDB51.

The exact choice of the framework used for finetuning influences the final accuracies substantially; an apples-to-apples comparison between different methods is impossible. For this reason, we additionally provide retrieval results. Here, the pretrained network serves as a feature extractor and is kept fixed. We extract features for all videos in the dataset and compute R@k: For each video in the test set we retrieve the top k nearest neighbor and count a correct retrieval if at least one of the videos is of the same class as the test video. Note that R@k does not measure the precision of the retrieved results. Therefore, we also present precision-recall-curves. Here, we compute precision and recall for all values of k and plot the resulting curves. Precision and recall are calculated as it is the standard approach in retrieval. Precision is the fraction of relevant instances among the retrieved instances, while recall is the fraction of relevant instances that were retrieved. For details see Section 3.2.2.

Finally, we evaluate our models on another transfer learning task: action segmentation. We use the pretrained model to extract features from the video frames of the Breakfast dataset and subsequently train a temporal action segmentation model on top of them. Again, this evaluation does not involve any finetuning on the target dataset. This provides a more elaborate assessment of the learned representations. We evaluate the segmentation model via frame-wise accuracy, segmental edit distance and F1 scores at overlapping thresholds 10%, 25% and 50%. More specifically, for the F1 scores we determine for each predicted action segment whether it is a true or false positive by taking a threshold on the IoU with the ground truth. Then we compute precision and recall summed over all classes and compute $\text{F1} = 2\frac{prec \cdot recall}{prec + recall}$.

6.3.2 Action Recognition

The most widely used framework for evaluating self-supervised representations utilizes the self-supervised pretrained weights to initialize a network, and then finetune it on a smaller annotated dataset. We consider the standard benchmarks UCF101 and HMDB51 and compare our method to previous self-supervised video representation methods in Table 6.1.

Since our method is based on RGB-video input only, we exclude multi-modal approaches such as (*Piergiovanni et al.*, 2020) and (*Patrick et al.*, 2021). The first block in Table 6.1 includes methods with shallower networks such as CaffeNet and significantly deeper architectures such as Resnet34 and S3D, and can therefore not be directly compared to our method. The second and third block includes methods using the same network architecture. Here, LSFD is trained with $N = 2$ and Sum aggregation.

While we outperform previous methods with a similar architecture on HMDB51, our method is inferior to the method of (*Jenni et al.*, 2020) on UCF101. Most notably, learning both stationary and non-stationary features with LSFD improves over our BFP method from Chapter 4 and our RINCE method from Chapter 5; this is especially evident on the HMMDB51 dataset. We attribute the smaller relative gain of our method on UCF101 to its inherently static bias (*Li et al.*, 2018; *Wang et al.*, 2019b), which is less pronounced on HMDB51. Moreover, note that since self-supervised learning is more relevant for smaller datasets like HMDB51, the results are more important compared with those on UCF101, which is a mid-sized dataset. For better comparisons, we additionally provide results with a C3D backbone (second block). Here, we outperform all previous methods.

Since the publication of our LSFD method, several other works have been published concurrently or subsequently, which outperform our method. For example, TimeEq (*Jenni and Jin*, 2021) has been published concurrently with LSFD and achieves very strong performance on UCF101 and HMDB51. They combine several self-supervised losses – some based on a contrastive loss, others on more traditional pretext task based losses – which together outperform our method by a large margin. This demonstrates the effectiveness of combining modern contrastive learning with established self-supervised learning from pretext tasks, and has the potential to be similarly effective for LSFD.

6.3.3 Ablation Studies

How much influence does each loss term have? Our LSFD method consists of three separate InfoNCE losses. We investigate the impact that each of them has on the resulting representation by progressively adding them to $\mathcal{L}_{\text{instance}}$; the results are provided in the first block of Table 6.2. Adding either $\mathcal{L}_{\text{stationary}}$ or $\mathcal{L}_{\text{non-stationary}}$ improves the performance on both UCF101 and HMDB51, suggesting that both stationary and non-stationary features are useful for action recognition. Note that the relative gain of adding $\mathcal{L}_{\text{non-stationary}}$ is higher on HMDB51 compared with $\mathcal{L}_{\text{stationary}}$. Adding both loss terms gives the highest performance, as is expected.

What is the impact of aggregation? The function used to aggregate non-stationary features in Eq. (6.2) plays a critical role in our proposed method. The simplest, non-parametric function we consider takes the Sum over non-stationary features. We also test parametric and increasingly more complex aggregation functions: a Linear mapping, an MLP, and a GRU. The results are provided in the second block of Table 6.2. Overall, we find that the simplest aggregation in form of a Sum yields the highest performance. Using a non-parametric aggregation function puts more load on the backbone, obligating it to do the heavy lifting, whereas parametric aggregation functions relax the task, allowing a potential shortcut via changing the impact of each ϕ_i. We keep the Sum aggregation for the remaining experiments. This observation is in line with learning via equivariances (*Lenc and Vedaldi*, 2015). Essentially, the aggregation function we apply on the feature space can be viewed as a complex transformation of splitting the input data along the temporal axis. A good representation

Loss	Agg	N	top1 Accuracy UCF101	HMDB51
$\mathcal{L}_{\text{instance}}$	Sum	2	72.7	46.3
$+\mathcal{L}_{\text{stationary}}$	Sum	2	74.4	48.7
$+\mathcal{L}_{\text{non-stationary}}$	Sum	2	74.8	51.6
all	Sum	2	77.2	**53.7**
all	Linear	2	77.1	51.3
all	MLP	2	75.7	49.6
all	GRU	2	75.5	51.0
all	Sum	3	77.8	52.1
all	Sum	4	**78.0**	52.3

Table 6.2: **Ablation Study on Loss Terms, Aggregation Function and the Number of Sub-sequences.** We ablate the different components of our method, including the different loss terms and several design choices, *i.e.* aggregation function (Agg) and number of sub-sequences N via finetuning on UCF101 and HMDB51. We see an inverse relation between the aggregation function complexity and performance of the learned representation on the downstream task. Performance of our method is marginally effected for larger values of N on HMDB51.

space should translate this complex transformation, *i.e.* mapping the short views to the long views via simple operations such as Sum. A similar discussion is conducted in (*Noroozi et al.*, 2017).

How many sub-sequences should we use? We ablate the effect of different numbers of sub-sequences N have on the learned representations in the third block of Table 6.2. We observed that training LSFD with $N = 3$ and $N = 4$ from scratch is sub-optimal; the task becomes increasingly difficult for larger N, decreasing the performance on both UCF101 and HMDB51, see Table 6.3. Therefore, we follow a curriculum learning strategy, that uses the pretrained model trained with $N - 1$ sub-sequences to initialize training for N sub-sequences. While increasing N improves the representation on UCF101, we observe a drop on HMDB51 in Table 6.2.

The effect of our curriculum learning strategy is layed out in Table 6.3. Here, we use the pre-trained weights obtained from training LSFD with $N - 1$ to initialize the training for N. We train $N = 2$ from scratch for 100 epochs, and subsequently train $N = 3$ and $N = 4$ for 40 epochs with a reduced learning rate and weight decay of $1e - 4$ and $1e - 6$, respectively. As evident in Table 6.3, this approach improves the downstream performance compared with training from scratch.

6.3.4 Video Retrieval

Next, we evaluate our method on video retrieval. We follow the protocol of (*Xu et al.*, 2019): We use the pretrained network to extract convolutional features at the last layer for all videos in the training and test set. For each test video we retrieve the top k nearest neighbors from the training videos. For R@k results in Table 6.4, we count a correct retrieval if the k nearest neighbors contain at least one video of the same class.

N	Training	top1 Accuracy UCF101	HMDB51
2	scratch	77.2	53.7
3	scratch	75.5	49.6
4	scratch	76.5	50.9
3	curriculum	77.8	52.1
4	curriculum	78.0	52.3

Table 6.3: **Curriculum Learning for Larger Number of Sub-sequences.** We train LSFD using different numbers of sub-sequences N, either from scratch (scratch) or in a curriculum learning regime (curriculum). We notice that training from scratch is sub-optimal, compared to curriculum learning, which we attribute to the increased difficulty of the task for larger N.

Method	Architecture	UCF101 R@1	R@5	R@10	R@20	HMDB51 R@1	R@5	R@10	R@20
VCOP (*Xu et al.*, 2019)	R3D	14.1	30.3	40.4	51.1	7.6	22.9	34.4	48.8
VCP (*Luo et al.*, 2020)	R3D	18.6	33.6	42.5	53.5	7.6	24.4	36.3	53.6
MemDPC (*Han et al.*, 2020b)	2D3D-R18	20.2	40.4	52.4	64.7	7.7	25.7	40.6	57.7
SpeedNet (*Benaim et al.*, 2020)	S3D-G	13.0	28.1	37.5	49.5	-	-	-	-
PRP (*Yao et al.*, 2020)	R3D	22.8	38.5	46.7	55.2	8.2	25.8	38.5	53.3
Temp Trans (*Jenni et al.*, 2020)	3D-R18	26.1	48.5	59.1	69.6	-	-	-	
CoCLR (*Han et al.*, 2020a)	S3D-G	**53.3**	**69.4**	**76.6**	**82.0**	23.2	43.2	53.5	65.5
LFSD (Ours)	3D-R18	44.9	64.0	73.2	81.4	**26.7**	**54.7**	**66.4**	**76.0**

Table 6.4: **Video Retrieval Evaluation.** Comparison to other methods via nearest neighbor video retrieval on UCF101 and HMDB51.

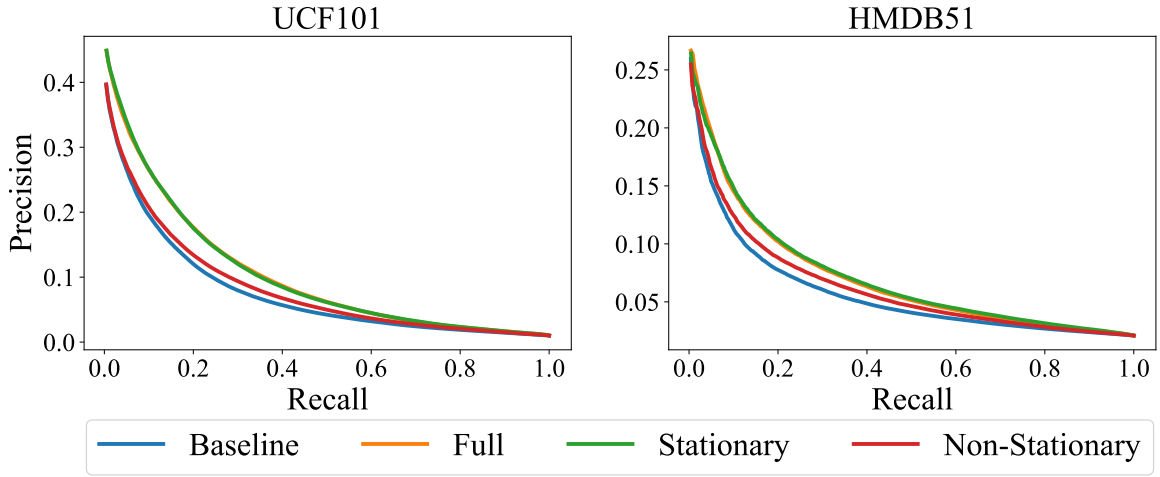


Figure 6.3: **Precision-Recall-Curves for UCF101 and HMDB51.** Stationary features are superior to non-stationary features on action recognition; both improve over our contrastive baseline.

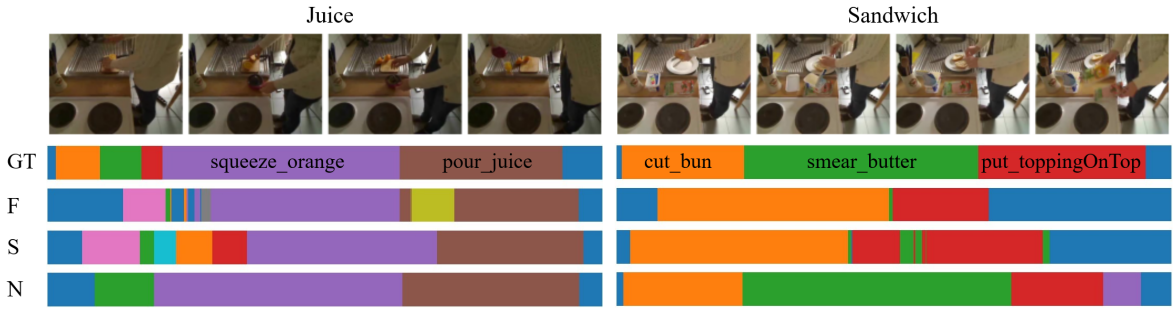


Figure 6.4: **Qualitative Evaluation.** Qualitative results of LSFD for two videos from the Breakfast dataset, showing the quality of stationary (S), non-stationary (N), and full (F) features for action segmentation. Non-stationary features provide higher quality representations than stationary features, validating their ability to capture fine-grained temporal variations.

While we improve over previous methods on HMDB51, our retrieval results on UCF101 are inferior to CoCLR (*Han et al.*, 2020a). Note that CoCLR uses a significantly deeper architecture. This is in line with our observation from the finetuning evaluation, where the relative improvement is higher on HMDB51 compared with UCF101. Note that this evaluation doesn't measure *precision* of the retrieved samples properly for $k > 1$. For this reason, we also present precision-recall curves in Figure 6.3 (for details see supplementary material). Our stationary features perform on-par with full features; non-stationary features slightly worse. This is even more pronounced on UCF101, where the static bias is higher (*Li et al.*, 2018). Both stationary and non-stationary features improve over our contrastive baseline on both datasets.

6.3.5 Temporal Action Segmentation

We evaluate the universal representation learning capability of our method via a temporal action segmentation downstream task. Given an untrimmed video, the goal of temporal action segmentation is to simultaneously segment every action in time and classify each obtained segment. Recent state-of-the-art action segmentation methods such as (*Farha and Gall*, 2019; *Li et al.*, 2020) train a temporal action segmentation model on top of pre-extracted features of the video frames. Usually, video frames are represented using deep 3D CNNs such as I3D (*Carreira and Zisserman*, 2017) pretrained on the Kinetics dataset, or hand-crafted features such as improved dense trajectories (IDT) (*Wang and Schmid*, 2013). In this experiment, we use the Breakfast dataset. This dataset consists of untrimmed videos containing fine-grained actions which are distinguishable mostly via temporal variations in the video, since the scene, actor, and objects remain similar throughout the video.

Using the Breakfast dataset allows us to better evaluate the temporal variation representation capability of our method. To that end, we use our frozen pretrained model as a feature extractor and compute features for each video frame of the datatset following (*Farha and Gall*, 2019). Then, we add a temporal action segmentation model on top, namely MS-TCN (*Farha and Gall*, 2019), and train for action segmentation in a fully supervised fashion. We use the official publicly available code of MS-TCN for training and evaluation. The MS-TCN model consists of four stages, each containing ten dilated convolutional layers. We train the model for 295 epochs using the Adam optimizer with an initial learning rate of 0.0005 and the `ReduceLROnPlateau` learning rate scheduler on the average

Method	F1@{10, 25, 50}			Edit	Acc
Random Initialization	39.3	32.4	21.8	41.2	32.6
Kinetics Supervised	47.1	41.7	31.0	54.5	45.1
$\mathcal{L}_{\text{instance}}$	44.6	39.9	31.4	50.3	57.9
LSFD, Full	46.1	41.7	32.6	56.3	60.1
LSFD, Stationary	40.6	35.7	28.8	54.7	58.9
LSFD, Non-Stationary	**52.0**	**42.8**	**35.3**	**60.0**	**60.6**

Table 6.5: **Action Segmentation Evaluation on the Breakfast dataset split 1.** We report results for several baselines and our LSFD method. We further split up the full feature in stationary and non-stationary features to investigate their effect. The backbone of all of the feature extraction models is 3D-ResNet18.

loss per epoch. The first layer of MS-TCN adjusts the dimension of the input features (*i.e.* 512 for full features and 256 for stationary and non-stationary features in our experiments) using a 1×1 convolution; the remaining layers have 64 channels.

As the segmentation model relies on the pre-extracted features, this evaluation reveals more reliably than finetuning how well our learned representations are suited for this downstream task that involves a better temporal understanding. All models in this section are based on a 3D-Resnet18 backbone that operates on RGB input only.

Evaluation metrics. For evaluation of the segmentation models, we report the frame-wise accuracy (Acc), segmental edit distance, and the segmental F1 score at overlapping thresholds 10%, 25% and 50% as proposed by *Lea et al.* (2017). While the frame-wise accuracy provides a basic rating, it is rather insensitive to over-segmentation errors and short action classes; longer action classes have a higher impact than short action classes. The segmental edit distance measures how well the model predicts the ordering of action classes, and is not impacted by the duration of action classes. The segmental F1 score measures the general quality of the segmentation model, as it penalizes over-segmentation and is also insensitive to the duration of the action classes. More specifically, we determine for each predicted action segment whether it is a true or false positive by taking a threshold on the IoU with the ground truth. Then we compute precision and recall summed over all classes and compute $\text{F1} = 2\frac{prec \cdot recall}{prec + recall}$.

Results. In Table 6.5 we provide the results for our unsupervised LSFD method as well as several baselines, including random initialization, full Kinetics supervision, and our contrastive baseline ($\mathcal{L}_{\text{instance}}$). Interestingly, we observe that all features obtained via unsupervised pretraining, *i.e.* the second block of the table, improve over a model trained with Kinetics supervision. This validates our argument that unsupervised learning may provide richer representations, capable of transferring better to different, unseen tasks. Furthermore, our method improves over the contrastive baseline as well as the supervised baseline by a significant margin, demonstrating that our long and short views enable a better temporal understanding. While the difference in accuracy is noticeable, it is even more prominent in the segmental edit distance and F1 scores, which better measure the overall

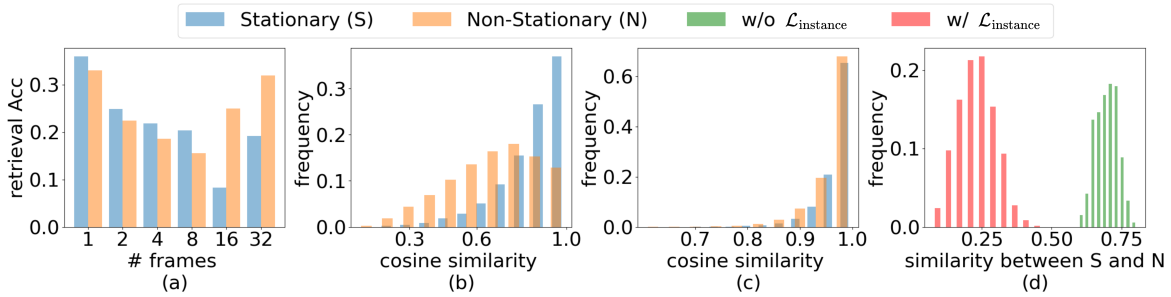


Figure 6.5: **Feature Decomposition Analyses.** (a) Retrieval accuracy of different subsets of HMDB51 based on the number of frames needed for classification. We observe that non-stationary features (N) outperform stationary features (S) on retrieving videos that require more frames (long views). The opposite happens for videos that require less frames (short views). Cosine similarities over time (b) with and (c) without $\mathcal{L}_{\text{instance}}$. We compute similarities of S and N features over time and show the histogram of the computed similarities on HMDB51. (b) S features are centered around high values when training with $\mathcal{L}_{\text{instance}}$; N features are distributed more uniformly. (c) When training without $\mathcal{L}_{\text{instance}}$ both S and N features are stable over time. (d) Similarities between S and N features when trained with and without $\mathcal{L}_{\text{instance}}$. We observe that removing $\mathcal{L}_{\text{instance}}$ results in a degenerate solution where S and N features are similar.

quality of the segmentations.

To investigate the feature decomposition we use only the stationary or only the non-stationary features of our pretrained model as input to the segmentation model. The quantitative results in Table 6.5 show that our non-stationary features outperform stationary features across all metrics, providing higher quality representations for the temporal segmentation model, and are consequently better suited for fine-grained temporal action segmentation. As can be seen in the raw frames in Figure 6.4, the static information across the video remains similar for most of the temporal segments. Hence, temporally segmenting such videos to fine-grained actions is a challenging problem requiring high quality features in terms of temporal variation representation. Moreover, the difference between the temporal segmentation performance of our stationary and non-stationary features also validates the feature space decomposition capability of our method. Additionally, we provide some qualitative results in Figure 6.4, where we observe a higher quality of non-stationary features compared with stationary features. As mentioned before, Breakfast is a challenging temporal action segmentation dataset due to the similarity of the objects, actor, and scene throughout the entire video. Therefore, the non-stationary features, which better represent the temporal variations in the video, empower the action segmentation model.

6.3.6 Feature Decomposition Analyses

The experiments above validate our hypothesis on feature decomposition by evaluating the performance of stationary and non-stationary features on dedicated downstream tasks. In the following, we conduct more specific analyses to obtain a better insight into our decomposed features. To that end, we use the pretrained model as a feature extractor (without the MLP heads h_s and h_n). To compute similarities between two feature vectors x and y, we use the cosine similarity, Eq. (3.4), $\frac{x^T y}{\|x\|\|y\|}$.

Furthermore, we compute retrieval accuracies among videos that can be classified with different

numbers of frames. First, we train separate model receiving $N = 1, 2, 4, 8, 16, 32$ frames as input. Then, we group the videos into disjoint subsets based on the numbers of frames that are needed for classification. For $N = 1$ this subset consists of all videos that are correctly classified by the model trained with $N = 1$ frames. For $N = 2$ the subset consists of those video that are correctly classified by the model with $N = 2$ frames as input, but that were misclassified by the $N = 1$ model, for $N = 4$ we exclude the videos from $N = 1$ and $N = 2$, etc.

Do we capture short- and long-term attributes? Figure 6.5(a) shows the retrieval accuracy of stationary and non-stationary features among the videos that can be classified with different numbers of frames. In each case, we exclude videos that are correctly classified with fewer frames, *i.e.* video that can be classified with a single frame are excluded from the set of videos that can be classified with two frames, etc. While stationary features achieve stronger retrieval accuracy on videos that can be classified with less frames (short views), *i.e.* less temporal context, non-stationary features are more beneficial for videos that require more frames (long views) and a longer temporal context.

Are stationary features more stable over time? We divide a given video into clips of 16 frames, compute the stationary and non-stationary features of all clips and similarities of the features over time. We show the histogram of the computed similarities on HMDB51 in Figure 6.5(b). Stationary features are centered around high values, while non-stationary features are distributed more uniformly. This suggests that stationary features remain more stable over time, whereas non-stationary feature vary.

What is the impact of $\mathcal{L}_{\text{instance}}$? Figure 6.5 (c) shows the same histogram as Figure 6.5 (b) for a network trained without $\mathcal{L}_{\text{instance}}$. We observe that in contrast to Figure 6.5 (b), stationary and non-stationary features behave similarly – both are stable over time. Moreover, we compare the histogram of similarities between stationary and non-stationary features of long views with and without $\mathcal{L}_{\text{instance}}$ in Figure 6.5 (d). Stationary and non-stationary features are very similar to each other when trained without $\mathcal{L}_{\text{instance}}$, suggesting a degenerate solution that copies stationary features as non-stationary ones. One reason why $\mathcal{L}_{\text{instance}}$ avoids this could be that it pushes the network towards exploiting the full capacity of the feature space, preventing redundant information in the full feature space.

6.4 Conclusion

In this chapter, we have introduced a novel method to decompose video representations into stationary and non-stationary features via contrastive learning from long and short views. We evaluated the learned representations extensively on multiple downstream tasks and datasets, and investigated various design choices and the role of stationary and non-stationary features. Overall, we find an interesting correlation between the type of features and the nature of the downstream task: Stationary features perform better on tasks and datasets with static biases, such as action recognition on UCF101, while non-stationary features are more beneficial for action segmentation, which requires better temporal understanding. We demonstrated a substantial gain in performance on the HMDB51 dataset for action recognition, and outperform a supervised baseline on the Breakfast dataset for action segmentation. We are the first to conduct an evaluation for video representation learning via

action segmentation, and there are two main takeaways: First, action segmentation provides an interesting and challenging downstream task to evaluate the quality of learned representations. Second, unsupervised video representation learning has the potential to improve representations used for action segmentation, and provides a promising approach for future research. We have found that a substantial improvement over supervised pretraining can be achieved. Furthermore, we analysed our feature decomposition and found that stationary features are more stable over time, while non-stationary features vary.

In the current and the previous two chapters, we introduced different methods for learning video representations in a self-supervised setting – each focusing on a different aspect. The BFP method in Chapter 4 introduces a set of hard negatives to focus on temporal features, while our RINCE method in Chapter 5 aims to reflect the gradually changing video content in the embedding space by utilizing temporally distant clips of the same video as hard negatives for temporally closer ones. In contrast to these two methods, which learn a single representation, LSFD aims to learn both stationary and non-stationary features. Furthermore, BPF and RINCE rely on (temporal) hard negatives in their contrastive losses, while LSFD does not explicitly incorporate any temporal or other hard negatives. Notably, the non-stationary contrastive loss of the LSFD method does not utilize any temporal negatives to encourage temporal features like the BFP method, but could potentially benefit from them as well. Therefore, a natural extension to the current approach would be to design a set of temporal hard negatives used in the non-stationary contrastive loss, *e.g.* by reversing or randomly shuffling the short views. Note that this approach requires an aggregation function that is not permutation invariant so that right and wrong temporal order can be distinguished. For example the aggregation functions considered in our ablation study, Linear, MLP, or GRU, are suitable for this purpose. Arguably, such temporal hard negatives should not be considered *true* negatives – ultimately they still show content that is highly related to the original sequence. Therefore, one could additionally consider to *rank* these temporal hard negatives using RINCE, *i.e.* considering the aggregated features of the correctly ordered short views as the higher ranked positives and the aggregated features of the incorrectly ordered short views as lower ranked positives.

CHAPTER 7

Temporal Action Segmentation via Sequence-to-Sequence Translation

In the previous chapters, we proposed different methods for self-supervised video representation learning based on contrastive learning, which enrich the learned representations and improve the performance on downstream tasks. We have seen in Chapter 6 that self-supervised methods have the potential to improve the representations for high-level video understanding tasks, such as temporal action segmentation, even compared with supervised pretraining. In this chapter, we will develop a method for temporal action segmentation, which operates on top of the pre-extracted features of a pretrained model. We identify a common challenge of current state-of-the-art action segmentation models – namely they are prone to *over-segmentation* due to the nature of their frame-wise predictions – and propose a novel method that views action segmentation from a sequence-to-sequence perspective, where we aim to predict the higher-level sequence of segments given the sequence of frames. We propose a Transformer-based model and incorporate several modifications and auxiliary loss functions. Furthermore, we extend our framework to the timestamp supervised setting with a constrained k-medoids algorithm. We evaluate our method empirically in both supervision settings on three challenging datasets for action segmentation. For fair comparisons to previous methods, the main experiments in this chapter are carried out with supervised pretrained I3D features, which are widely used in the literature. As indicated in Chapter 6, unsupervised pretrained features can potentially benefit the action segmentation task; therefore, we additionally evaluate our model on the LSFD pretrained features from the previous chapter.

Individual Contribution

The following chapter is based on the publication (*Behrmann et al.*, 2022):

Unified Fully and Timestamp Supervised Temporal Action Segmentation via Sequence to Sequence Translation
Nadine Behrmann*, S. Alireza Golestaneh*, Zico Kolter, Juergen Gall, and Mehdi Noroozi
European Conference on Computer Vision (ECCV), 2022.
doi: 10.1007/978-3-031-19833-5_4.
* S. Alireza Golestaneh and Nadine Behrmann are joint first authors.

This publication was done in close collaboration between S. Alireza Golestaneh, Mehdi Noroozi and Nadine Behrmann. Juergen Gall provided scientific guidance and very valuable feedback and suggestions. All authors contributed to the writing of the paper. The initial idea to view action segmentation from a sequence-to-sequence perspective and apply an auto-regressive Transformer model

was proposed by Nadine Behrmann and jointly implemented by S. Alireza Golestaneh and Nadine Behrmann. The modifications and auxiliary loss functions were developed in joint discussions between S. Alireza Golestaneh, Mehdi Noroozi and Nadine Behrmann. Namely, Nadine Behrmann proposed the frame-wise loss term, while Mehdi Noroozi proposed the cross-attention loss, and S. Alireza Golestaneh suggested the group-wise regularization loss terms as well as the modifications on the encoder architecture. To extend our method to the timestamp supervised setting, S. Alireza Golestaneh, Mehdi Noroozi and Nadine Behrmann jointly developed a constrained k-medoids algorithm to generate pseudo-segmentations.

Contents

7.1 Introduction

The ability to analyze, comprehend, and segment video content at a temporal level is crucial for many computer vision, video understanding, robotics, and surveillance applications. Recent state-of-the-art methods for action segmentation mainly formalize the task as a frame-wise classification problem; that is, the objective is to assign an action label to each frame, based upon the full sequence of video frames. We illustrate this general approach in Figure 7.1 (a). However, this formulation suffers several drawbacks, such as over-segmentation when trained on relatively small datasets (which typically need to consist of expensive frame-level annotations).

In this work, we propose an alternative approach to the action segmentation task. Our approach involves a Transformer-based seq2seq architecture that aims to map from the video frames directly to a *higher-level sequence* of action segments, *i.e.* a sequence of action label / duration pairs that describes the full predicted segmentation.

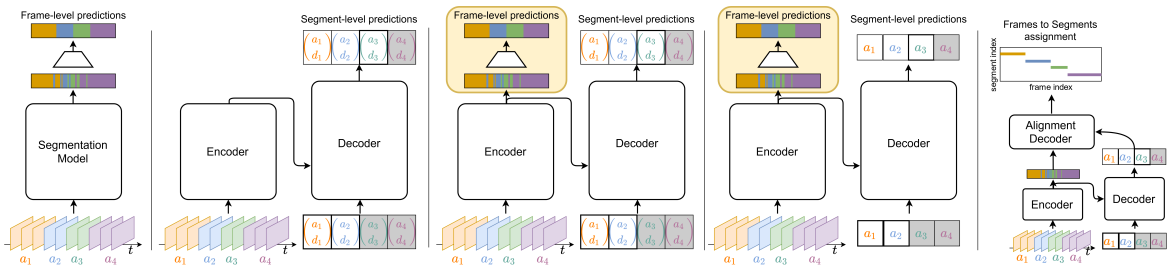


Figure 7.1: **Using Transformers for Action Segmentation.** Instead of frame-level predictions, which are prone to over-segmentation (a), we propose a seq2seq Transformer model for segment-level predictions (b). To provide more direct feedback to the encoder we apply a frame-wise loss (c); the resulting features enhance the decoder predictions. However, duration prediction still suffers, so we focus on transcript prediction (d) and use a separate alignment decoder to fuse encoder and decoder features to arrive at an implicit form of duration prediction (e).

The basic structure of our model follows traditional Transformer-based seq2seq models: the encoder branch takes as input a sequence of video frames and maps them to a set of features with the same length; the decoder branch then takes these features as input and generates a predicted sequence of high-level action segments in an auto-regressive manner. This approach, illustrated in Figure 7.1 (b), is a natural fit for action segmentation because it allows the decoder to directly output sequences in the higher-level description space. The main advantage over the frame-level prediction is that it is less prone to over-segmentation.

However, this seemingly natural approach does not immediately perform well on the action segmentation task by itself. In contrast to language translation, action segmentation typically involves long input sequences of very similar frames opposed to short output sequences of action segments. This difference together with the relatively small amount of training videos, makes it challenging for the encoder and decoder to keep track of the full information flow that is necessary to predict the high-level segmentation alone. For this reason, we incorporate several modifications and additional loss terms into our system, which together make this approach compete with or improve upon the state-of-the-art.

First, to provide more immediate feedback to the encoder, we employ a frame-wise loss that linearly classifies each frame with the corresponding action label given the encoder features, Figure 7.1 (c). As a result, the encoder performs frame-wise classification with high localization performance, *i.e.* high frame-wise accuracy, but low discrimination performance, *i.e.* over-segmentation with low Edit distance to the ground truth. Nonetheless, its features provide the decoder an informative signal to predict the sequence of actions more accurately. This immediate auxiliary supervision signal allows the decoder to learn more discriminative features for different actions. While the frame-wise loss improves the transcript prediction, the decoder still suffers from low localization performance for duration prediction. As the next step, we fuse the decoder predictions with the encoder, for which we propose two solutions. First, we propose to fuse the discriminative features of the decoder with the encoder features via a cross-attention mechanism in an alignment decoder, Figure 7.1 (d)+(e). Second, the high performance of our decoder on predicting transcripts and the high

performance of our encoder on localizing actions allows us to effectively utilize the common post-processing algorithm such as FIFA (*Souri et al.*, 2021a) and Viterbi (*Richard et al.*, 2018; *Kuehne et al.*, 2020).

Finally, we further extend our proposed framework when only a weaker form of timestamp supervision is available. As mentioned before, the frame-wise prediction is vital in our Transformer model to cope with small datasets and long sequences of frames. In this case, when the frame-level annotations are not fully available, we assign a label to each frame by a constrained k-medoids clustering algorithm that takes advantage of timestamp supervision. Our simple proposed clustering method achieves a frame-wise accuracy of up to 81% on the training set, which can be effectively used to train our seq2seq model. We further show that the clustering method can also be used in combination with frame-wise prediction methods such as ASFormer (*Yi et al.*, 2021).

We evaluate our model on three challenging action segmentation benchmarks: 50Salads (*Stein and McKenna*, 2013), GTEA (*Fathi et al.*, 2011), and Breakfast (*Kuehne et al.*, 2014). While our method achieves competitive frame-wise accuracies compared to the state-of-the-art, our method substantially outperforms other approaches in predicting the action sequence of a video, which is measured by the Edit distance. By using Viterbi (*Richard et al.*, 2018; *Kuehne et al.*, 2020) or FIFA (*Souri et al.*, 2021a) as post-processing, our approach also achieves state-of-the-art results in terms of segmental F1 scores. To the best of our knowledge, this work is the first that utilizes Transformers in an auto-regressive manner for action segmentation and is applicable to both the fully and timestamp supervised setup.

7.2 Method

In this section, we provide the details for our proposed **U**nified **V**ideo **A**ction **S**egmentation model via **T**ransformers (**UVAST**). The goal of temporal action segmentation is to temporally segment long, untrimmed videos and classify each of the obtained segments. Current state-of-the-art methods are based on *frame-level* predictions – they assign an action label to each individual frame – which are prone to *over-segmentation*: The video is not accurately segmented into clean, continuous segments, but fragmented into many shorter pieces of alternating action classes. We challenge this view of frame-level predictions and propose a novel approach that directly predicts the segments instead. By focusing on *segment-level* predictions – an alternative but equivalent representation of segmentations – our method overcomes the deep-rooted over-segmentation problem of frame-level predictions.

7.2.1 Transformer for Auto-Regressive Segment Prediction

In this work, we view action segmentation from a sequence-to-sequence (seq2seq) perspective: mapping a sequence of video frames to a sequence of action segments, *e.g.* as pairs of action label and segment duration. The Transformer model (*Vaswani et al.*, 2017) has emerged as a particularly powerful tool for seq2seq tasks and may seem like the natural fit. The vanilla Transformer model consists of an encoder module that captures long-range dependencies within the input sequence and a decoder module that translates the input sequence to the desired output sequence in an auto-regressive manner. In contrast to language translation tasks, action segmentation faces a strong mismatch between input and output sequence lengths, *i.e.* inputs are long and untrimmed videos with various sequence lengths, while outputs are relatively shorter sequences of action segments. Therefore, we incorporate

several modifications to address these issues, which we will go over in more detail in the following.

Notation. Given an input sequence of T frame-wise features z_t, for frame $t \in \{1, \dots, T\}$, our goal is to temporally segment and classify the T frames. The ground-truth labels of a segmentation can be represented in two equivalent forms: 1) a sequence of frame-wise action labels $\tilde{y}_t \in \mathcal{C}$ for frame t, where $\mathcal{C}$ is the set of action classes, 2) a sequence of segment-wise annotations, which consists of ground-truth segment action classes $\tilde{c}_i \in \mathcal{C}$ (also known as *transcript*), and segment lengths $\tilde{l}_i \in \mathbb{R}_+$ for each segment $i \in \{1, \dots, K\}$.

Transformer Encoder. Our input sequence $Z \in \mathbb{R}^{T \times d}$ consists of T frame-wise features z_t, where d denotes the features dimension. We embed them using a linear layer, and then feed them to the Transformer encoder, which consists of several layers, and allows the model to capture long-range dependencies within the video via the self-attention mechanism. The output of the encoder, $E \in \mathbb{R}^{T \times d'}$, is a sequence of frame-wise features e_t, which will be used in the cross-attention module of the decoder. In order to provide direct feedback to the encoder, we apply a linear layer to obtain frame-level predictions for e_t. This enables the encoder to accurately localize the action classes within the video and provides more informative features to the decoder. In practice, we use a modified version of the encoder proposed in (*Yi et al.*, 2021), which locally restricts the self-attention mechanism and uses dilated convolutions; details on the exact architecture and modifications are provided in Section 3.3.3.

Transformer Decoder. Given a sequence of frame-wise features $E \in \mathbb{R}^{T \times d'}$, we use a Transformer decoder to auto-regressively predict the transcript, *i.e.* the action labels of the segments. Starting with a *start-of-sequence (sos)* token we feed the sequence of segments $S \in \mathbb{R}^{K \times d'}$ – embedded using learnable class tokens and positional encoding – up until segment i to the decoder. Via the cross-attention between the current sequence of segments and frame-wise features, the decoder determines the next segment $i + 1$ in the video. In principle, the decoder could predict the segment durations as well (Figure 7.1 (c)), however, in practice we found that the decoder's duration prediction suffers from low localization performance, see Table 7.4. While it is sufficient to pick out a single or few frames in the cross-attention mechanism for predicting the correct action class of a segment, the duration prediction is more difficult since it requires to assign frames to a segment and count them. Since the number of segments is much smaller than the number of frames, the cross-attention mechanism tends to assign only a subset of the frames to the correct segment. To address this issue, we propose a separate decoder module, which fuses the discriminative decoder features with the highly localized encoder features to obtain a more accurate duration prediction, which we describe in Section 7.2.3.

7.2.2 Training Objective

Although our ultimate goal is segment-level predictions, we provide feedback to both the encoder and decoder model to make the best use of the ground-truth labels. To that end, we apply a frame-wise cross-entropy loss on the frame-level predictions of the encoder:

$$\mathcal{L}_{\text{frame}} = -\frac{1}{T} \sum_{t}^{T} \log(y_{t,\hat{c}}), \tag{7.1}$$

where $y_{t,c}$ denotes the predicted probability of label c at time t, and $\hat{c}$ denotes the ground-truth label of frame t. Analogously, we apply a segment-wise cross-entropy loss on the segment-level predictions of the decoder:

$$\mathcal{L}_{\text{segment}} = -\frac{1}{K}\sum_{i}^{K}\log(c_{i,\hat{c}}), \tag{7.2}$$

where $c_{i,c}$ denotes the predicted probability of label c at segment i, and $\hat{c}$ denotes the ground-truth label of segment i.

Regularization via Grouping. To regularize the encoder and decoder predictions, we additionally apply *group-wise* cross-entropy losses. To that end, we group the frames and segments by ground-truth labels $L = \{c \in \mathcal{C} | c \in \{\tilde{c}_1, \ldots, \tilde{c}_K\}\}$ that occur in the video: $T_c = \{t \in \{1, \ldots, T\} | \tilde{y}_t = c\}$ are the indices of frames with class c, and $K_c = \{i \in \{1, \ldots, K\} | \tilde{c}_i = c\}$ the indices of segments with class c. We apply a cross-entropy loss to the averaged prediction of each group:

$$\mathcal{L}_{\text{g-frame}} = -\frac{1}{\|L\|}\sum_{c \in L}\log\left(\frac{1}{\|T_c\|}\sum_{t \in T_c} y_{t,c}\right) \tag{7.3}$$

$$\mathcal{L}_{\text{g-segment}} = -\frac{1}{\|L\|}\sum_{c \in L}\log\left(\frac{1}{\|K_c\|}\sum_{i \in K_c} c_{i,c}\right) \tag{7.4}$$

7.2.3 Cross-Attention Loss

We utilize a loss on the cross-attention mechanism between the encoder and decoder features to allow further interactions between them. Let us assume that T video frames and corresponding K actions in the encoder and decoder are represented by their features $E \in \mathbb{R}^{T \times d'}$ and $D \in \mathbb{R}^{K \times d'}$, respectively. The cross-attention loss involves obtaining a cross-attention matrix $M = \texttt{softmax}(\frac{ED^T}{\tau'\sqrt{d'}})$, where τ' is a stability temperature, and each row of M includes a probability vector that assigns each encoder feature (frame) to decoder features (actions). We then use M in the following cross-entropy loss function:

$$\mathcal{L}_{\text{CA}}(M) = -\frac{1}{T}\sum_{t}\log(M_{t,\hat{k}}), \tag{7.5}$$

where $\hat{k}$ is the ground-truth segment index to which frame t belongs. We use this loss in our transcript decoder (main decoder) and alignment decoder in the following.

Cross-Attention Loss for the Transcript Decoder. The cross-attention loss, when applied to the transcript decoder, provides more intermediate feedback to the decoder about the action location in the input sequence, see Figure 7.4. We found this loss term especially effective on smaller datasets such as 50Salads (see Table 7.5). Our main objective to train the encoder and the transcript decoder is:

$$\mathcal{L} = \mathcal{L}_{\text{frame}} + \mathcal{L}_{\text{segment}} + \mathcal{L}_{\text{g-frame}} + \mathcal{L}_{\text{g-segment}} + \mathcal{L}_{\text{CA}}(M). \tag{7.6}$$

Cross-Attention Loss for the Alignment Decoder. While the transcript decoder generates the sequence of actions in a video, it does not predict the duration of each action. Although it is possible to predict the duration as well, as illustrated in Figure 7.1 (c), the transcript decoder still struggles to localize actions through direct duration prediction as shown in Table 7.4. One reason for this could be the high mismatch between input and output sequence length and the relatively small number of training videos. While picking up a single segment frame is sufficient to predict the action class, the duration prediction effectively requires counting the number of frames in the segment, resulting in a more challenging task. Therefore, we design an alternative alignment decoder for predicting segment durations implicitly.

A high Edit score of our decoder indicates that it has already learned discriminative features of the actions. The motivation for our alignment decoder is to align the encoder features to the highly discriminative features of the decoder, which can be further used for the duration prediction (see Fig 7.1 (e)). In essence, our proposed alignment decoder is a one-to-many mapping from the decoder features to the encoder features. The alignment decoder takes the encoder and decoder features $E \in \mathbb{R}^{T \times d'}$ and $D \in \mathbb{R}^{K \times d'}$ with positional encoding as input and generates the aligned features $A \in \mathbb{R}^{T \times d'}$. Since the alignment decoder aims to explore the dependencies between the encoder features and the decoder features, we employ a cross-attention mechanism in its architecture similar to the transcript decoder. To this end, we compute an assignment matrix $\overline{M} \in \mathbb{R}^{T \times K}$ via cross-attention between the alignment decoder features (A) and positional encoded features of the transcript decoder (D) by $\overline{M} = \texttt{softmax}(\frac{AD^T}{\tau})$ with a small value of τ. Note that with a small value of τ each row of $\overline{M}$ will be close to a one-hot encoding indicating the segment index the frame is assigned to. The positional encoding for D resolves ambiguities if the same action occurs at several locations in the video.

In contrast to the decoder from the previous section, the alignment decoder is not auto-regressive since the full sequences of frame-wise and segment-wise features are already available from the previous encoder and decoder. During inference, we compute the segment durations by taking the sum over the assignments:

$$l_i = \sum_t \overline{M}_{t,i}, \tag{7.7}$$

where $i \in \{1, ..., k\}$ and $\overline{M}_{t,i}$ denotes whether frame t is assigned to segment i. We found that training the alignment decoder using only the loss Eq. (7.7) for $\overline{M}$ in a separate stage on top of the frozen encoder and decoder features results in a more robust model that suffers less from overfitting.

7.2.4 Timestamp Supervision

In this section, we show how our proposed framework can be extended to the timestamp supervised setting. In this setting, we are given a single annotated frame for each segment in the video, *i.e.,* frame annotations are reduced dramatically, and ground-truth segment durations are no longer available for all frames. As we extensively discussed before, our proposed framework relies on the frame-level supervisory signal on top of the encoder. However, it turns out that a noisy frame-level annotation provides a solid signal to the encoder. To obtain such frame-level annotations, we propose a constrained k-medoids algorithm that propagates the timestamp supervision to all frames.

A typical k-medoids algorithm starts with random data points as the cluster centers. It iteratively updates the cluster centers chosen from the data points and the assignments based on their similarity

Algorithm 1: Constrained K-Medoids to generate temporally continuous clusters.

Input: T features z_t, timestamps $[t_1, \dots, t_k]$
Init: $m_i = z_{t_i}$ `# initialize medoids`
repeat
  $D_{i,j} = \text{dist}(m_i, z_j)$ `# pairwise costs`
  $b_0 = 0; b_n = T$ `# compute boundaries`
  for $i = 1, \dots, k-1$ **do**
    $b_i = \text{argmin}_l(\sum_{j=t_i}^{l} D_{i,j} + \sum_{j=l}^{t_{i+1}} D_{i+1,j})$
  end
  for $i = 1, \dots, k$ **do**
    $t_i = \text{argmin}_l(\sum_{j=b_{i-1}+1}^{b_i} \text{dist}(z_l, z_j))$
    $m_i = z_{t_i}$ `# new medoids`
  end
until *until convergence*;
return $l_i = b_i - b_{i-1}$

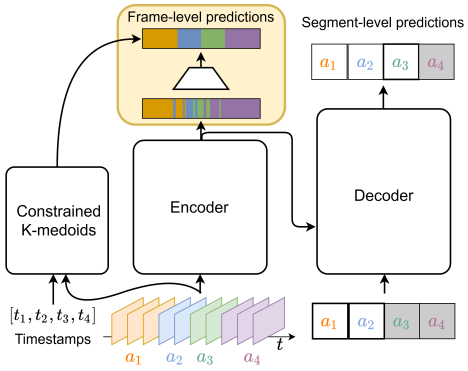


Figure 7.2: **Constrained K-Medoids.** Given frame-wise features and timestamps, our constrained k-medoids algorithm generates a pseudo-segmentation that guides the encoder during the training instead of ground truth frame-level labels in a fully supervised setup.

to the cluster center. Having access to the timestamp supervision, we can use them as initialization and cluster the input features. However, in a standard k-medoids algorithm, a temporally continuous set of clusters are not taken for granted. We call our method constrained k-medoids because we force the clusters to be temporally continuous. This can be simply achieved by modifying the assignment step of the k-medoids algorithm. Instead of assigning pseudo-labels to each frame, we find the temporal boundaries of each cluster. In the assignment step, we update the boundaries such that the accumulative distance of each cluster to the current center is minimized. Alg. 1 summarizes the steps of our clustering method. In principle, we can apply k-medoids using the frame-wise input features z_t, the encoder features e_t, or a combination of both. In practice, we found that using input features alone gives surprisingly accurate segmentations, see Table 7.9 and Section 7.3.9 for more analyses.

7.3 Experiments

7.3.1 Datasets

We evaluate the performance of our proposed model extensively on three challenging action segmentation datasets (50Salads (*Stein and McKenna*, 2013), GTEA (*Fathi et al.*, 2011), and Breakfast (*Kuehne et al.*, 2014)). Among the aforementioned datasets, Breakfast is the largest publicly available dataset for action segmentation in terms of the number of classes and videos. In all of our experiments, we follow previous work (*Ishikawa et al.*, 2021; *Yi et al.*, 2021; *Li et al.*, 2020; *Farha and Gall*, 2019; *Wang et al.*, 2020c; *Chen et al.*, 2020a) and perform 4-fold cross-validation on Breakfast and GTEA and 5-fold cross-validation on 50Salads.

7.3.2 Evaluation Metrics

All evaluation metrics are introduced in Section 3.2.3; here, we briefly recapitulate the most important properties of each metric. For evaluation, following previous works, we report the frame-wise accuracy (Acc), segmental edit score (Edit), and the segmental F1 score at overlapping thresholds 10%, 25%, and 50%, denoted by F1@$\{10, 25, 50\}$ (*Lea et al.*, 2017). The overlapping threshold is determined based on the intersection over union (IoU) ratio. Although frame-wise accuracy is the most commonly used metric for action segmentation, it does not portray a thorough picture of the performance of action segmentation models. A major disadvantage of frame-wise accuracy is that long action classes have a higher impact than short action classes and dominate the results. Furthermore, over-segmentation errors have a relatively low impact on Acc, which is particularly problematic for applications such as video summarization. On the other hand, Edit and F1 scores establish more comprehensive measures of the quality of the segmentations (*Lea et al.*, 2017); Edit measures the quality of the predicted transcript of the segmentation, while F1 scores penalize over-segmentation and are also insensitive to the duration of the action classes. Our proposed method performs particularly well on the Edit and F1 scores on all datasets and in fully and timestamp supervised setups, achieving state-of-the-art results in most cases.

7.3.3 Implementation Details and Training

We follow the standard training strategy from existing algorithms (*Farha and Gall*, 2019; *Li et al.*, 2020; *Ishikawa et al.*, 2021; *Yi et al.*, 2021; *Singhania et al.*, 2021) and train our main network

(Section 7.2.1) end-to-end with batch size of 1. We train our model for at most 800 epochs using the Adam optimizer (*Kingma and Ba*, 2015) with learning rate 0.0005 and the loss in Eq. (7.6). In the cross-attention loss, Eq. (7.5), we set $\tau = 1$ to ensure training stability, and $\tau = 0.0001$ during inference. As input for our model, we use the same I3D (*Carreira and Zisserman*, 2017) features that were used in many previous works; these features consist of 2048-dimensional feature vectors extracted from RGB videos and optical flow and are projected to 64-dimensional feature vectors in the first layer of the model. Similar to previous methods and to have a fair comparison, we use sampling rate 1 for the Breakfast and GTEA dataset and a sampling rate of 2 for the 50Salads dataset since it has higher FPS compared with the other two datasets.

Due to a strong imbalance in the duration of action segments, we propose a *split-segment* approach for improved training: longer action segments are split up into several shorter ones so that segment durations are more uniformly distributed; for details and ablations, see Section 7.3.8. During the inference, we do not use any split-segment and use the entire video. Furthermore, we use random dropping of the features as an augmentation method during training, where we randomly drop $\sim 1\%$ of the features in the sequence.

During inference we optionally use FIFA or Viterbi to compute the segment durations. For Viterbi decoding we use a frame sampling rate of 5 for the Breakfast dataset, 2 for 50Salads, and 1 for GTEA. We run FIFA for 3000 epochs using a sharpness parameter of 80 and step size of 0.01 for Breakfast and 50Salads and 0.1 for GTEA.

For the timestamp supervised setting, we use the same training hyper-parameters as above and substitute ground truth segmentations with the pseudo-segmentations generated with Algorithm 1. All of the training and testing experiments were conducted on a single NVIDIA V100 GPU.

Network Architecture. For the encoder architecture, we used a modified version of the encoder proposed in (*Yi et al.*, 2021), namely we replace the `ReLU` activation functions with `GeLU` and add another layer of dilated temporal convolutions, for details see Section 3.3.3. The impact of our modifications is shown in Section 7.3.8. Our encoder model consists of 10 layers. For the decoder, we use a standard decoder architecture (*Vaswani et al.*, 2017), with two layers and single head attention. Furthermore, we use a cross-attention smoothing (average pooling on the cross-attention map along the T dimension using a kernel size of 31) for the 50Salads dataset to reduce the noise in the cross-attention. Compared to the Breakfast and GTEA datasets, the 50Salads dataset contains the longest videos (dataset statistics can be found in Table 2.2) with only a few training samples, which causes the cross-attention map to be noisy. Our proposed cross-attention loss along with cross-attention smoothing helps to reduce the noise and leads to a better performance.

For the alignment decoder (Section 7.2.3), we use a single layer, single head decoder. To train this model, we use similar hyper-parameters and optimizers while freezing the encoder-decoder model from Section 7.2.1 and only train the alignment decoder with our cross-attention loss. For positional encoding, we use the standard sinusoidal positional encoding (*Vaswani et al.*, 2017).

7.3.4 Fully Supervised Results

Here, we provide the overall performance comparison of our proposed method, UVAST, on three challenging action segmentation datasets with different levels of supervision. We demonstrate the effectiveness of our proposed method for both the fully supervised and timestamp supervised setup

and achieve competitive results on both settings. We provide the results of our proposed model for four scenarios: Transcript prediction of our encoder-decoder architecture (referred to as "w/o duration") and three different approaches to obtain durations for the segments, namely, the alignment decoder from Section 7.2.3 ("+ alignment decoder"), Viterbi ("+ Viterbi"), and FIFA ("+ FIFA"). We only report the Edit score for "w/o duration", as it does not provide segment durations. Next, the three approaches for computing durations include a learnable module – our alignment decoder from Section 7.2.3 that predicts durations implicitly – and two inference algorithms where we leverage the transcripts and frame-wise predictions to compute the durations, namely Viterbi decoding and FIFA (*Souri et al.*, 2021a). A significant advantage of our method is that a predicted transcript is readily available and can be used in these inference algorithms in contrast to previous methods, which need to iterate over the training transcripts. Furthermore, we can optionally use the predicted duration of the alignment decoder to initialize the segment durations in FIFA.

Table 7.1 shows the performance of our method in the fully supervised setting compared with state-of-the-art fully supervised methods. At the bottom of Table 7.1 we provide the results of our proposed model for the four scenarios explained above. UVAST achieves significantly better Edit score on transcript prediction ("w/o duration") than all other existing methods on all three datasets, which demonstrates the effectiveness of our model to capture and summarize the actions occurring in the video. In the last three rows of Table 7.1, we use three different approaches to compute the durations of the segments. Combining UVAST with the alignment decoder from Section 7.2.3 achieves competitive results. However, it is important to note that Transformers are very data-hungry and training them on the small dataset can be challenging. We observe that UVAST with alignment decoder outperforms other methods in terms of Edit score. While the F1 scores are comparable to the state-of-the-art on the Breakfast dataset, the small size of the GTEA dataset hinders the training of the alignment decoder.

Method		Breakfast					50Salads					GTEA				
		F1@{10, 25, 50}			Edit	Acc	F1@{10, 25, 50}			Edit	Acc	F1@{10, 25, 50}			Edit	Acc
TDRN (*Lei and Todorovic*, 2018)		-	-	-	-	-	72.9	68.5	57.2	66.0	68.1	79.2	74.4	62.7	74.1	70.1
SSA-GAN (*Gammulle et al.*, 2020)		-	-	-	-	-	74.9	71.7	67.0	69.8	73.3	80.6	79.1	74.2	76.0	74.4
MuCon (*Souri et al.*, 2021b)		73.2	66.1	48.4	76.3	62.8	-	-	-	-	-	-	-	-	-	-
DTGRM (*Wang et al.*, 2021)		68.7	61.9	46.6	68.9	68.3	79.1	75.9	66.1	72.0	80.0	87.3	85.5	72.3	80.7	77.5
Gao *et al.* (*Gao et al.*, 2021)		74.9	69.0	55.2	73.3	70.7	80.3	78.0	69.8	73.4	82.2	89.9	87.3	75.8	84.6	78.5
MS-TCN++ (*Li et al.*, 2020)		64.1	58.6	45.9	65.6	67.6	80.7	78.5	70.1	74.3	83.7	88.8	85.7	76.0	83.5	80.1
BCN (*Wang et al.*, 2020c)		68.7	65.5	55.0	66.2	70.4	82.3	81.3	74.0	74.3	84.4	88.5	87.1	77.3	84.4	79.8
SSTDA (*Chen et al.*, 2020a)		75.0	69.1	55.2	73.7	70.2	83.0	81.5	73.8	75.8	83.2	90.0	89.1	78.0	86.2	79.8
Singhania *et al.* (*Singhania et al.*, 2021)		70.1	66.6	56.2	68.2	73.5	76.6	73.0	62.5	69.2	80.1	90.5	88.5	77.1	87.3	**80.3**
ASRF (*Ishikawa et al.*, 2021)		74.3	68.9	56.1	72.4	67.6	84.9	83.5	77.3	79.3	84.5	89.4	87.8	79.8	83.7	77.3
ASFormer (*Yi et al.*, 2021)		76.0	70.6	57.4	75.0	73.5	85.1	83.4	76.0	79.6	85.6	90.1	88.8	79.2	84.6	79.7
ASFormer (*Yi et al.*, 2021) + Viterbi		76.1	70.5	57.1	74.5	70.2	84.1	82.3	74.9	76.1	84.7	91.1	90.0	79.5	86.5	80.0
ASFormer (*Yi et al.*, 2021) + FIFA		76.8	71.4	**58.9**	75.6	**73.7**	84.5	83.2	75.4	78.5	85.4	90.4	88.6	78.1	86.2	78.9
UVAST (Ours)	w/o duration	-	-	-	76.9	-	-	-	-	**83.9**	-	-	-	-	**92.1**	-
	+ alignment decoder	76.7	70.0	56.6	**77.2**	68.2	86.2	81.2	70.4	**83.9**	79.5	77.1	69.7	54.2	90.5	62.2
	+ Viterbi	75.9	70.0	57.2	76.5	66.0	**89.1**	**87.6**	**81.7**	**83.9**	**87.4**	**92.7**	**91.3**	**81.0**	**92.1**	80.2
	+ FIFA	**76.9**	**71.5**	58.0	77.1	69.7	88.9	87.0	78.5	**83.9**	84.5	82.9	79.4	64.7	90.5	69.8

Table 7.1: **Fully Supervised Temporal Action Segmentation Results.** Best and second best results are shown in bold and underlined, respectively. Our method outperforms state-of-the-art in terms of Edit distance on all datasets, and in terms of F1 scores on the Breakfast and 50Salads datasets.

Moreover, with frame-wise predictions and transcript prediction available, our method conveniently allows applying inference algorithms at test time, such as FIFA and Viterbi, without the need

to expensively iterate over the training transcripts. Combining our method with Viterbi outperforms the existing methods on GTEA and 50Salads in terms of Edit and F1 scores, and achieves competitive results on Breakfast. We also provide the results of UVAST with FIFA, where we initialize the duration with the predicted duration. It achieves strong performance on Breakfast and 50Salads. Note that although FIFA is a fast approximation of Viterbi, it achieves better results on the Breakfast dataset. This is due to the fact that the objective function that is minimized by FIFA/Viterbi does not optimize the evaluation metrics directly, *i.e.* the global optimum of the Viterbi objective function does not guarantee the global optimum of the evaluation metrics. This observation is consistent with the results reported in (*Souri et al.*, 2021a).

The comparison to ASFormer (*Yi et al.*, 2021) is also interesting. While ASFormer – like most other approaches – performs frame-level predictions, Figure 7.1 (a), UVAST predicts the action segments in an auto-regressive manner, Figure 7.1 (d)+(e). As expected, ASFormer achieves in general a better frame-wise accuracy, while UVAST achieves a better Edit score. Since ASFormer uses a smoothing loss and multiple refinement stages to address over-segmentation similar to MS-TCN (*Farha and Gall*, 2019; *Li et al.*, 2020), it has $\sim$ 1.3M learnable parameters, whereas our proposed model has $\sim$ 1.1M parameters. Our approach with Viterbi achieves similar F1 scores on the Breakfast dataset, but higher F1 scores on the other datasets. For a more thorough and fair comparison with ASFormer, we additionally provide the results when combining ASFormer with Viterbi or FIFA during inference. To that end, we extract the transcript to be used in Viterbi/FIFA from the frame-wise predictions of the model.

Overall, we find that our method achieves strong performance in terms of Edit and F1 scores, while Acc is compared to the state-of-the-art lower on Breakfast. Note that Acc is dominated by long segments and less sensitive to over-segmentation errors. Lower Acc and higher Edit/F1 scores indicate that UVAST localizes action boundaries, which are difficult to annotate precisely, less accurately. It is therefore an interesting research direction to improve the segment boundaries, *e.g.* by using an additional refinement like ASFormer.

7.3.5 Timestamp Supervision Results

We use our proposed constrained k-medoids to generate pseudo-segmentation using the frame-wise input features and ground truth timestamps. The output consists of continuous segments, which can be identified with the transcript to yield a pseudo-segmentation. While this approach can be applied both to the input features and encoder features in principle, we find that using the input features already gives a surprisingly good performance, see Table 7.9. Note that this is not a temporal action segmentation method as it requires timestamp supervision as input. We use the resulting pseudo-segmentation as the auxiliary signal to our encoder during training where we have access to the timestamp supervision.

In Table 7.2, we compare our proposed timestamp model with the recently proposed method (*Li et al.*, 2021c) on the three action segmentation datasets. To the best of our knowledge, (*Li et al.*, 2021c) is the first work that proposed and applied timestamp supervision for the temporal action segmentation task. Although other weakly supervised methods exist, they are based on *transcript* supervision, a weaker form of supervision; therefore, we additionally train MS-TCN (*Farha and Gall*, 2019) and ASFormer (*Yi et al.*, 2021) with our constrained k-medoids. To get more thorough and fair comparisons, we further show their performance when combined with Viterbi decoding or

Method		Breakfast					50Salads					GTEA				
		F1@{10, 25, 50}			Edit	Acc	F1@{10, 25, 50}			Edit	Acc	F1@{10, 25, 50}			Edit	Acc
Li et al. (*Li et al.*, 2021c)		70.5	63.6	47.4	69.9	**64.1**	73.9	70.9	60.1	66.8	75.6	78.9	73.0	55.4	72.3	66.4
MS-TCN (*Farha and Gall*, 2019)		56.1	50.0	36.8	61.7	62.5	74.4	70.4	57.7	66.8	72.8	82.8	80.3	63.5	79.5	67.7
MS-TCN (*Farha and Gall*, 2019) + Viterbi		43.3	37.2	25.6	43.5	35.9	74.0	70.0	55.5	68.2	72.8	82.6	79.7	61.6	81.0	68.1
MS-TCN (*Farha and Gall*, 2019) + FIFA		36.1	30.5	21.9	42.6	35.8	73.7	69.5	54.9	69.2	72.8	81.7	77.7	57.7	81.0	67.3
ASFormer (*Yi et al.*, 2021)		70.9	62.9	44.0	71.6	61.3	76.6	72.1	59.6	70.0	76.9	**87.2**	83.1	67.5	83.0	68.8
ASFormer (*Yi et al.*, 2021) + Viterbi		71.3	63.1	44.3	71.1	60.7	76.3	72.1	59.4	68.8	**77.0**	87.1	83.1	**68.2**	83.0	69.1
ASFormer (*Yi et al.*, 2021) + FIFA		71.1	62.7	44.3	71.8	61.8	76.7	72.0	58.8	70.0	76.9	86.8	81.9	65.4	83.0	68.4
UVAST (Ours)	+ alignment decoder	**72.0**	64.1	**48.6**	**74.3**	60.2	75.7	70.6	58.2	78.4	67.8	70.8	63.5	49.2	88.2	55.3
	+ Viterbi	71.3	63.3	48.3	74.1	60.7	**83.0**	**79.6**	**65.9**	78.2	**77.0**	**87.2**	**83.7**	66.0	**89.3**	**70.5**
	+ FIFA	**72.0**	**64.2**	47.6	74.1	60.3	80.2	74.9	61.6	**78.6**	72.5	80.7	75.2	57.4	88.7	66.0

Table 7.2: **Timestamp Supervision Results for Temporal Action Segmentation.** UVAST, ASFormer (*Yi et al.*, 2021), and MSTCN (*Farha and Gall*, 2019) are trained via our constrained k-medoids pseudo-labels. Best result shown in bold. UVAST outperforms SOTA on all datasets and metrics except for Acc on Breakfast. The performance in terms of Edit distance is significant, and is comparable to the fully supervised setup.

FIFA during inference.

Table 7.2 shows that: I) our method largely outperforms the other methods by achieving the best performance on 13 out of 15 metrics. Analogously to the fully supervised case, we observe the strong performance of our alignment decoder in terms of Edit and F1 scores on Breakfast; with FIFA and Viterbi, we outperform the method of (*Li et al.*, 2021c) on 50Salads and GTEA. Notably, UVAST achieves significantly higher performance in terms of Edit distance, which is comparable to the fully supervised setup. II) ASFormer and MSTCN perform reasonably well in the timestamp supervision setup when trained on the pseudo-labels of our constrained k-medoids algorithm, which demonstrates once more the effectiveness of our proposed constrained k-medoids algorithm. III) ASFormer and MSTCN do not benefit from the Viterbi algorithm in this case. This is due to the relatively lower Edit distance of these methods. Namely, Viterbi hurts MSTCN on Breakfast as it achieves significantly lower Edit distance compared to ours.

7.3.6 Combining UVAST with LSFD Features

In Chapter 6, we have introduced a self-supervised video representation learning method for learning stationary and non-stationary features. While the standard evaluation protocol for video representation learning mainly focuses on the performance of action recognition downstream tasks, we argued that other downstream tasks should be considered for a more thorough evaluation. We consequently proposed an action segmentation downstream task, which allows us to better evaluate our model's ability to represent temporal variations. In Chapter 6, we evaluated our LSFD method by training the popular MS-TCN (*Farha and Gall*, 2019) method on top of the features extracted using an LSFD pretrained model. The MS-TCN method is – like most current action segmentation methods – based on frame-wise predictions, and while the multi-stage architecture and smoothing loss of MS-TCN helps to reduce over-segmentation errors, the model can still be relatively sensitive to them.

Therefore, we additionally evaluate the pretrained features from the previous chapter with the UVAST model. One the one hand, this allows us to compare UVAST and MS-TCN in a more nuanced way, when different types of pretrained features are used as input to the models. On the other hand,

	Features	F1@{10, 25, 50}			Edit	Acc
MS-TCN	Kinetics Supervised	47.1	41.7	31.0	54.5	45.1
	$\mathcal{L}_{\text{instance}}$	44.6	39.9	31.4	50.3	57.9
	LSFD, Full	46.1	41.7	32.6	56.3	60.1
	LSFD, Stationary	40.6	35.7	28.8	54.7	58.9
	LSFD, Non-Stationary	52.0	42.8	35.3	60.0	60.6
UVAST	Kinetics Supervised	61.6	55.7	41.2	65.4	48.9
	$\mathcal{L}_{\text{instance}}$	63.8	58.4	45.2	66.9	52.2
	LSFD, Full	65.0	58.7	46.7	67.2	55.3
	LSFD, Stationary	65.5	59.9	47.4	67.1	53.1
	LSFD, Non-Stationary	66.7	61.8	48.0	68.9	57.0

Table 7.3: **UVAST on LSFD Features.** We train UVAST + alignment decoder on the different pretrained features from Chapter 6 on split 1 of the Breakfast dataset; this includes features extracted using a Kinetics supervised model, a model pretrained with an instance recognition loss $\mathcal{L}_{\text{instance}}$, and a model pretrained with our LSFD method from Chapter 6. The first block evaluates the features using an MS-TCN model and corresponds to the results reported in Table 6.5. The second block evaluates the same features using our UVAST model with the alignment decoder. In comparison to MS-TCN, UVAST achieves higher F1 and Edit scores across all features. Similar to Chapter 6, we find that the unsupervised pretrained features largely outperform Kinetics supervised features on this task; furthermore, the non-stationary features perform best.

it allows us to further evaluate the LSFD method and its different types of features. To that end, we evaluate the same pretrained features used in Table 6.5, in Chapter 6. This includes a model pretrained with Kinetics supervision (corresponding to the standard supervised pretraining paradigm) as well as several unsupervised pretrained representations: A model pretrained with traditional contrastive learning, *i.e.* an instance recognition loss $\mathcal{L}_{\text{instance}}$, and the different features learned with our LSFD method. All methods were pretrained using the same 3D-Resnet18 architecture (*Hara et al.*, 2018) with RGB video as input. In contrast, the features used for the main experiments in this chapter were pretrained with Kinetics supervision using an I3D backbone (*Carreira and Zisserman*, 2017) with RGB and optical flow input. Note that the I3D architecture is significantly deeper than the 3D-Resnet18 and further utilizes additional optical flow input, enabling the model to learn more powerful representations compared with a 3D-Resnet18 network. Therefore, the experiments in this section can not directly be compared to the main experiments in Table 7.1.

We report the results in Table 7.3. The first block corresponds to the experiments in Table 6.5; the second block corresponds to UVAST trained on top of the different features discussed above. First, we observe that UVAST outperforms MS-TCN in terms of Edit and F1 scores by substantial margins across all different types of feature inputs. This again demonstrates the advantage of predicting action segmentations at the segment-level rather than frame-level as it is much less prone to over-segmentation errors. Second, we observe that UVAST is much less sensitive to the type of feature input and achieves comparable performance across the different self-supervised representation learning methods. The difference between stationary and non-stationary or full features is much less pronounced here, while all of them outperform the InfoNCE ($\mathcal{L}_{\text{instance}}$) based features. Similar to

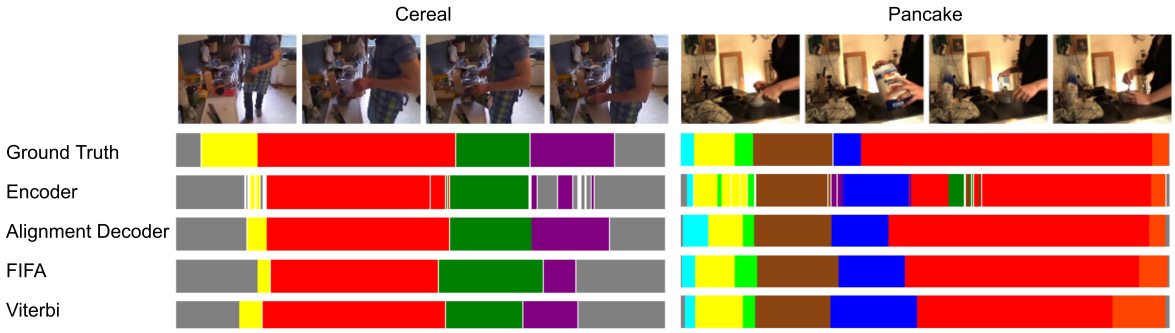


Figure 7.3: **Qualitative Results.** We show ground truth and predicted segmentation of fully supervised (left) and timestamp supervised (right) UVAST of two videos from the Breakfast dataset.

the evaluation with MS-TCN, we observe slightly better performance of the non-stationary features over stationary and full features. Third, similar to Chapter 6, we find that the self-supervised learned features outperform the Kinetics supervised learned features, which again demonstrates the potential of self-supervised pretraining for this task. However, for a more conclusive assessment whether self-supervised pretraining is superior to supervised pretraining, a comparison with a deeper backbone architecture, additional optical flow input and potentially longer training would be required. As such experiments are computationally very demanding, we leave them for future work and conclude here that for relatively smaller architectures such as 3D-Resnet18 self-supervised pretraining is favorable.

7.3.7 Qualitative Evaluation

We show qualitative results of two videos from the Breakfast dataset in the fully supervised and timestamp supervised setting in Figure 7.3. We visualize the ground truth segmentations (first row) as well as the predicted segmentations of our encoder (second row) and decoder with alignment decoder, FIFA or Viterbi for duration prediction (last three rows). The encoder predictions demonstrate well the common problem of over-segmentation with frame-level predictions; the segment-level predictions of our decoder on the other hand yield coherent action segments.

7.3.8 Ablations Studies

Duration Prediction. As discussed in Section 7.1, the vanilla Transformer model, Figure 7.1 (b), does not generalize to the action segmentation task, see Table 7.4. We train this model using $\mathcal{L}_{\text{segment}}$ and the MSE between predicted and ground truth durations. To that end, we rescale the absolute durations to $[0, 1]$ by dividing the number of frames in a segment by the total number of frames T in the video. Our first modification involves applying a frame-wise loss the the encoder features, which drastically improves the results. However, this explicit form of duration prediction still struggles to accurately localize the segments. Predicting duration implicitly via our alignment decoder instead, Figure 7.1 (d)+(e), on the other hand improves the localization, increasing Acc and F1.

Impact of the Loss Terms. In Table 7.5 we investigate the impact of the different loss terms (Section 7.2.2) and their combinations on split 1 of the Breakfast and 50Salads datasets. In the first row of Table 7.5, we evaluate the encoder when trained only using the frame-wise loss. As expected,

Method	Loss	F1@{10, 25, 50}			Edit	Acc
Vanilla Transformer, Figure 7.1 (b)	Eq. (7.2) + MSE	48.1	42.3	26.7	52.9	35.0
+ Frame-wise Loss, Figure 7.1 (c)	Eq. (7.2) + Eq. (7.1) + MSE	70.7	63.5	44.4	73.9	59.1
	Eq. (7.6) + MSE	72.1	65.1	48.7	76.5	59.0
+ Alignment Decoder, Figure 7.1 (d)+(e)	Eq. (7.2) + Eq. (7.1) + AD	73.5	68.3	54.3	75.2	67.7
	Eq. (7.6) + AD	77.1	72.0	60.4	78.2	71.7

Table 7.4: **Explicit Duration Prediction on Breakfast split 1.** We show the results of different steps described in Section 7.1 from explicit duration prediction via the vanilla Transformer to implicit duration prediction with our alignment decoder and the impact of the full loss function Eq. (7.6).

solely relying on the frame-wise loss leads to over-segmentation and poor performance. The remaining rows of Table 7.5 show the performance of our proposed model when using both encoder and decoder as explained in Sections 7.2.1 and 7.2.2, and reflect the key idea of our method to directly predict the segments. While the most basic version using the segment-wise loss, Eq. (7.2), improves over frame-wise predictions, we observe that using both frame-wise, Eq. (7.1), and segment-wise, Eq. (7.2), loss term increases the performance drastically. Moreover, we observe that adding the cross-attention loss. Eq. (7.5), further improves the results, demonstrating the effectiveness of this loss for longer sequences with many action segments, such as 50Salads. While adding the group-wise loss terms, Eqs. (7.3) and (7.4), individually improves the performance moderately, the real benefit is revealed when combining them all together.

	Breakfast				50Salads			
	F1@{10, 25, 50}			Edit	F1@{10, 25, 50}			Edit
$\mathcal{L}_{\text{frame}}$	8.9	7.7	5.9	14.1	13.5	12.8	10.8	11.4
$\mathcal{L}_{\text{segment}}$	49.5	39.7	22.9	55.6	20.1	16.3	8.6	29.2
$\mathcal{L}_{\text{frame}}$+$\mathcal{L}_{\text{segment}}$	71.8	66.3	52.6	73.4	55.0	52.4	37.5	45.3
$\mathcal{L}_{\text{frame}}$+$\mathcal{L}_{\text{segment}}$+$\mathcal{L}_{\text{CA}}$	73.8	67.0	54.8	74.5	74.2	71.0	58.4	65.5
$\mathcal{L}_{\text{frame}}$+$\mathcal{L}_{\text{segment}}$+$\mathcal{L}_{\text{g-frame}}$	73.3	65.8	52.8	73.6	56.6	53.4	40.2	44.0
$\mathcal{L}_{\text{frame}}$+$\mathcal{L}_{\text{segment}}$+$\mathcal{L}_{\text{g-segment}}$	72.8	64.3	53.7	73.2	59.1	56.1	42.8	51.6
$\mathcal{L}_{\text{frame}}$+$\mathcal{L}_{\text{segment}}$+$\mathcal{L}_{\text{g-frame}}$+$\mathcal{L}_{\text{g-segment}}$	73.5	67.9	55.0	73.1	57.0	54.5	40.4	42.4
$\mathcal{L}_{\text{frame}}$+$\mathcal{L}_{\text{segment}}$+$\mathcal{L}_{\text{g-frame}}$+$\mathcal{L}_{\text{g-segment}}$+$\mathcal{L}_{\text{CA}}$	75.1	68.9	54.9	76.1	73.6	71.5	55.3	78.4

Table 7.5: **Loss Ablation.** Contribution of different loss terms on Breakfast and 50Salads (split 1).

Impact of the Cross-Attention Loss. To shed more light on the contribution of our cross-attention loss we visualize its impact in Figure 7.4. Given the ground truth segmentation, Figure 7.4 (a), of a video from the 50Salads dataset, Figure 7.4 (b) shows our expected target activations (output of the softmax) of the decoder's cross-attention map; we hypothesize that activations should be higher in areas that belong to the corresponding segment. Figure 7.4 (c) shows the output of the cross-attention when using our proposed cross-attention loss. We observe that this loss indeed guides the cross-attention to have higher activations in the regions that belong to the related segment of an

action. Figure 7.4 (d) shows that lack of our cross-attention loss causes the attention map to be noisy; it is unclear which region is used for the segment classification. Furthermore, as shown in Table 7.5 the lack of our cross-attention loss degrades the performance more on datasets with long sequences and many segments (*e.g.* 50Salads dataset).

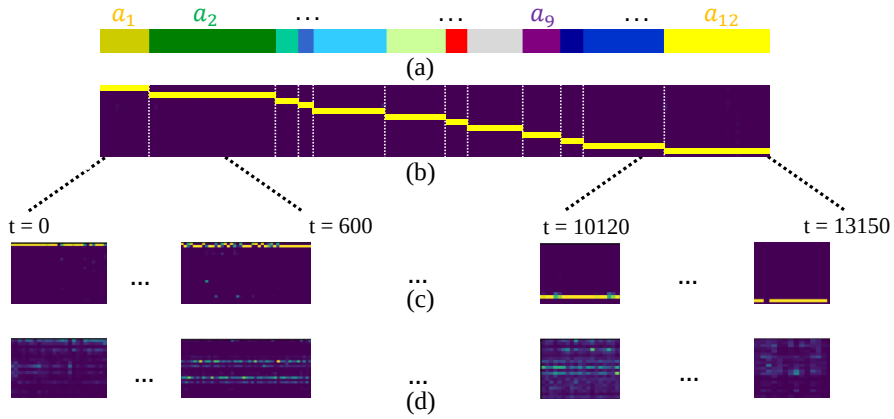


Figure 7.4: **Impact of the Cross-Attention Loss for the Transcript Decoder.** (a) A ground truth example of a video with 13150 frames and 12 segments from the 50Salads dataset. (b) The target cross-attention map after softmax with dimension 12×13150. (c) and (d) show the zoomed-in segments of the cross-attention map of the decoder when using the cross-attention loss (top) or not using it (bottom). In (b-d) brighter color means higher values of the activations.

Impact of Split-Segment. Due to a strong imbalance in the duration of action segments we propose a *split-segment* approach for improved training, where longer action segments are split up into several shorter ones, so that segment durations are more uniformly distributed; it is important to note that during the inference we *do not* use any split-segment and use the video as is. In the timestamp supervised setting we do not use split-segment as we do not have access to the ground truth duration of segments. For instance, if the split-segment value is set to 0.1 it means that the normalized duration of each action segment should be at most 0.1 and segments larger than 0.1 will be split up into smaller segments with maximum length of 0.1. During inference we merge the repeated actions into one: For example, if our model predicts an action sequence of (A, B, B, C, A, A, A) we convert it to (A, B, C, A). The split-segment value is a hyper parameter that can be selected empirically for each dataset. In Table 7.6 we show the impact of our split-segment approach on split 1 of the Breakfast, 50Salads, and GTEA datasets, where we report the Edit scores. Our split-segment approach helps the model achieve better performances on all three datasets. Furthermore, in Table 7.7 we provide an ablation study regarding the impact of selecting different split-segment values on the split 1 of the Breakfast dataset.

Impact of the Encoder Model. As mentioned above, we utilize a modified version of the encoder model proposed in (*Yi et al.*, 2021). The encoder model proposed by *Yi et al.* (2021) takes advantage of window attention (*i.e.* by restricting the attention mechanism to a local window) as well as hierarchical representations for the action segmentation task. We made a few small modifications to the architecture that further improved the performance; details can be found in Section 3.3.3.

	Breakfast	50Salads	GTEA
Without Split-Segment	75.0	74.2	88.2
With Split-Segment	76.1	75.4	93.4

Table 7.6: **Impact of Split-Segment.** Quantitative comparison (Edit score) between the impact of using split-segment versus not using it on split 1 of GTEA, 50Salads, and Breakfast dataset.

	Split-Segment Values					
	0.05	0.1	0.15	0.17	0.2	0.3
Breakfast	74.9	75.6	76.1	76.1	76.1	75.4

Table 7.7: **Impact of Split-Segment Values.** Ablation study on split 1 of the Breakfast dataset regarding the impact of different split-segment values on the performance (Edit score).

Specifically, we replace the `ReLU` activation layers with `GeLU` activation and add one more dilated convolution layer at the end of each encoder block. In Table 7.8, we report the Edit score when using a simple encoder block (*Vaswani et al.*, 2017), the encoder proposed by *Yi et al.* (2021), and our modified version (last row in Table 7.8) on split 1 of GTEA, 50Salads, and Breakfast. While the ASFormer encoder achieves drastic improvements over the simple encoder, our modified version further improves over the ASFormer encoder.

		Breakfast	50Salads	GTEA
UVAST with	Simple Encoder	70.3	68.7	69.4
	ASFormer (*Yi et al.*, 2021) Encoder	74.6	73.8	88.6
	Proposed Encoder	76.1	75.4	93.4
UVAST Timestamp	ASFormer (*Yi et al.*, 2021) Encoder	72.9	76.6	89.0
	Proposed Encoder	74.3	78.3	89.1

Table 7.8: **Impact of the Encoder Model.** Quantitative comparison (Edit score) between the impact of different encoder blocks on split 1 of GTEA, 50Salads, and Breakfast.

7.3.9 K-Medoids

In Section 7.2.4, we propose a constrained k-medoids algorithm for generating pseudo-segmentations given frame-wise input features and timestamps. In contrast to the vanilla k-medoids clustering algorithm, our constrained version ensures temporal consistency of the clusters and the resulting temporally continuous clusters can be unambiguously identified with the class labels of the ground truth transcript. This is a major advantage over an unconstrained clustering method, which may result in temporally fragmented clusters, making class label assignment ambiguous. We compare our constrained k-medoids with the unconstrained version, see Table 7.9, where we assign each cluster the class label belonging to the timestamp it was initialized with. Note, that in this scenario the original ground truth timestamps may end up in completely different clusters and the class label

	Breakfast					50Salads					GTEA				
	F1@{10, 25, 50}			Edit	Acc	F1@{10, 25, 50}			Edit	Acc	F1@{10, 25, 50}			Edit	Acc
Constrained k-medoids	95.5	87.5	70.0	100.0	75.4	97.5	90.4	75.6	100.0	81.3	99.8	97.7	83.0	100.0	75.3
Unconstrained k-medoids	8.4	6.6	3.8	12.3	53.8	3.8	2.5	1.0	2.3	52.9	71.0	68.2	52.9	59.8	69.6

Table 7.9: **Constrained vs. Unconstrained K-Medoids.** We evaluate the pseudo-segmentations generated with our constrained k-medoids algorithm, Algorithm 1, given the frame-wise input features and ground truth timestamps. We compare our constrained k-medoids algorithm with the vanilla unconstrained version.

assignment becomes noisy.

As expected the temporal fragmentation of the clusters leads to over-segmentation and correspondingly low Edit and F1 scores. However, even on Acc this unconstrained version suffers due to the noisy label assignment. Furthermore, we show two example videos from the Breakfast dataset in Figure 7.5; again, we observe over-segmentation due to temporally fragmented clusters. Notably, we observe that the unconstrained k-medoids performs much better on GTEA compared with Breakfast and 50Salads. One reason for this can be the frequency of background classes, which are visually distinct to the action classes in the video and typically show very static scenes. The frequent background classes of highly similar features separate the action classes from one another. In contrast, Breakfast and 50Salads do not have a frequent background class and relatively static scenes are assigned to an action class, making it more difficult to separate them.

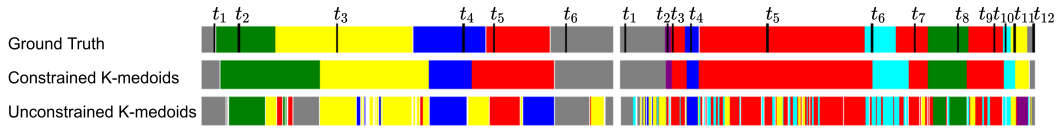


Figure 7.5: **Qualitative Evaluation of K-Medoids.** We compare our constrained and the unconstrained k-medoids algorithm qualitatively. Both algorithms cluster the frame-wise input features, using the ground truth timestamps $t_1, \ldots, t_k$ as initialization.

7.4 Conclusion

In this chapter, we have introduced UVAST, a new unified framework for fully and timestamp supervised temporal action segmentation. Here, we view action segmentation from a sequence-to-sequence perspective, *i.e.* mapping a sequence of frames to a sequence of action segments, and propose a Transformer-based model that directly predicts the segments. While the segment-level predictions of our model addresses the over-segmentation problem effectively, this new design entails a new challenge: predicting the duration of segments explicitly does not work out-of-the-box. Therefore, we proposed three different approaches to alleviate this problem, enabling our model to achieve competitive performance on all three datasets; we expect future work to successfully improve it even further. We evaluated our method both quantitatively and qualitatively on three different datasets with varying levels of supervision and thoroughly investigated the different components of our model.

As end-to-end training on the long untrimmed videos is extremely computationally demanding, action segmentation models are typically built on top of pre-extracted frame-wise features , *i.e.* video representations. While the standard approach for this purpose is still supervised pretraining, we demonstrated that self-supervised video representation learning has the potential to improve the representations for this task in Chapter 6. We further evaluated the learned representations from Chapter 6 by training our UVAST model on top and validate the findings: The different self-supervised features outperform Kinetics supervised ones across all metrics. However, a direct comparison of these self-supervised learned features to the widely used supervised I3D features is not possible as we use a significantly less powerful 3D-Resnet18 backbone trained on RGB input only. Therefore, the question whether these findings extend to deeper network architectures remains open and provides an interesting direction for future work.

CHAPTER 8

Conclusion

Contents

8.1 Overview

In this thesis, we have addressed video understanding problems that can broadly be categorized into two areas: Video representation learning on the one hand and temporal action segmentation on the other. The first concerns the task of accurately representing videos in terms of high-level abstract concepts, that are useful for a variety of different downstream tasks, without the need for labelled data. To that end, we have developed several methods to improve self-supervised video representation learning, specifically via contrastive learning. A common challenge of video understanding models is that they struggle to accurately represent temporal variations in videos. To encourage more temporally structured representations, we developed a bidirectional feature prediction (BFP) task in Chapter 4 that leverages unobserved past video clips in addition to future clips. Given a present video clip, the model not only needs to predict the features of past and future clips, but also distinguish between them. This approach relies on the traditional binary contrastive learning framework, which only distinguishes between two categories of pairs: positives and negatives. However, this strict binary separation can be limiting in cases, where samples can not be clearly categorized as positive or negative, *e.g.* temporally distant clips have been considered as either positives or negatives in the literature. To properly incorporate such samples, we propose a ranking InfoNCE loss (RINCE) that not only allows for but also benefits from a ranked definition of positives. Both BFP and RINCE, like most video representation methods in the literature, aim to learn a single video representation. The features in videos can be very diverse, including some that remain the same over time (stationary features) and others that vary (non-stationary features), and representing them in a single representation may be sub-optimal. Therefore, we decompose the representation into stationary and non-stationary

features in Chapter 6, which improves the performance on several downstream tasks, including a temporal action segmentation task. Here, we use the learned video representation model to extract a set of frame-wise features, which we use as input to an action segmentation model that is trained on top. This not only allows us to evaluate the learned representation more thoroughly, but also demonstrates the importance of video representation learning for this task and addresses action segmentation from one angle. On the other hand, action segmentation also requires more high-level temporal reasoning, which is typically performed by the action segmentation model that is built on top of the video representations. In Chapter 7, we target action segmentation from this perspective and develop a sequence-to-sequence Transformer model. The recent state-of-the-art action segmentation models are largely based on frame-wise predictions, which are inherently prone to over-segmentation errors. In contrast to these frame-wise prediction models, we propose a fundamentally different approach and aim to directly map to the higher-level sequence of action segments.

8.2 Summary of Contributions

The overall goal of this thesis was to improve self-supervised video representations, specifically with a focus on temporal features in videos, and develop a method for temporal action segmentation that directly predicts the action segments. In the following, we will review and summarize the main contributions of this thesis.

8.2.1 Leveraging Past Frames for Video Representations

As discussed in Section 1.2.2, temporal features in videos are important for different video understanding tasks – yet they can be difficult to learn in a fully supervised setting. Therefore, Research Question 2 investigates how we can learn temporal features in videos, specifically without the need for large-scale labelled datasets. Addressing this question, we developed a self-supervised method for learning temporally structured video representations in Chapter 4. A popular approach for self-supervised video representation learning derives a learning signal based on future prediction; given a set of present frames the model is asked to predict the contents of future frames of some form. However, such methods overlook a different, complementary set of frames: unobserved past. In Chapter 4, we demonstrate how both future and past prediction can be jointly incorporated in a contrastive learning framework to enhance the learned representations. How to exploit the complementary supervision signal from past frames effectively proves to be non-trivial and we investigate different approaches. Ultimately, we find a joint past and future feature prediction task that incorporates temporal hard negatives to be most effective. To that end, we aggregate the past and future features to obtain a joint past-future representation, which serves as the positive sample in the contrastive loss; given the present frames, the model predicts the joint past-future features. To encourage more temporally structured representations, we incorporate the reverse order of future and past as a temporal hard negative. This effectively requires the model not only to predict the past and future but also to distinguish between them. In practice, both past and future frames show actions and scenes that are highly related to the present frames. Therefore, treating past prediction in the same manner as future prediction does not necessarily stimulate the model to learn a different set of features. In contrast, our hard temporal negatives force the model to additionally identify which actions have happened *before* the given present video sequence and which *after*.

8.2.2 Ranking Positive Samples in Contrastive Learning

Traditional contrastive learning requires a strict binary separation of samples into positive and negative pairs. In practice, this might not always be possible or desirable. For example, consider several temporally distinct clips of the same video. With increasing temporal distance between clips their content likely becomes more different – should temporally distant clips be considered as positive or negative samples? In fact, this is still an open research question and methods in both directions have been proposed. To properly incorporate such samples, we propose a method that allows for a ranked definition of positives with varying levels of similarity in Chapter 5. Given a ranked set of positives, our model aims to mirror the similarity ranking in the feature space. To encourage gradually decreasing similarity for lower ranked positives, we apply a series of InfoNCE losses, one for each rank. Here, the higher ranked positives utilize lower ranked positives as hard negatives, ensuring the positives of the current rank are more similar to the query than the lower ranked positives. This framework allows us to rank video clips based on their temporal distance to the query clip, resulting in higher-quality video representations.

8.2.3 Decomposing Representations into Stationary and Non-Stationary Features

As outlined in Section 1.1 and 3.1.1, an ideal video representation should benefit many diverse downstream tasks and enable video understanding models to answer many complex questions about what happens in the video. Such diverse tasks in turn demand a diverse set of features. While our method in Chapter 4 is more focused on representing the temporal structure of videos, we recognize the importance of both stationary and non-stationary features for different downstream tasks in Chapter 6. Stationary features, which remain similar throughout the video, enable the prediction of video-level attributes, such as action classes. Non-stationary features on the other hand, represent temporally varying features and are more beneficial for downstream tasks involving more fine-grained temporal features, such as temporal action segmentation. Learning such diverse features in a single representation may be sub-optimal; therefore, we decompose the representation space into stationary and non-stationary features.

In order to learn stationary and non-stationary features, we use contrastive learning from long and short views, *i.e.* long video sequences and their shorter sub-sequences. Here, we design the positive pairs in separate contrastive losses in a manner to promote the learning of stationary and non-stationary features. As stationary features remain similar over time, they are shared between long and short view and we construct the positive pairs in the stationary contrastive loss accordingly. The non-stationary features, on the other hand, vary over time, *i.e.* the non-stationary features of the long view encompass an aggregated form of the non-stationary features of the sequence of short views. We design the positive pair in the non-stationary contrastive loss in alignment with this observation. We evaluate our method extensively and find that, in fact, the stationary features are especially beneficial for action recognition downstream tasks, while the non-stationary features benefit an action segmentation downstream task.

8.2.4 Sequence-to-Sequence Translation for Temporal Action Segmentation

Video representation learning addresses the task how to extract descriptive features from video frames, which constitutes the first step in many action segmentation methods. In a second step, an

action segmentation model is built on top of the sequence of framewise features. While the current methods treat action segmentation as a framewise classification problem, our Research Question 3 challenges this approach and investigates whether it is possible to directly map to a higher-level output space that describes the segmentation at a segment-level instead of frame-level. In Chapter 7, we demonstrate that this is indeed possible and develop a Transformer-based sequence-to-sequence model, which takes a sequences of framewise features and maps it to a sequence of action segments. As a vanilla Transformer model does not achieve a satisfactory performance, we propose a series of modifications and auxiliary losses to account for the strong mismatch between input and output sequence length. This includes a framewise classification loss on the output of the encoder to provide more direct feedback to the model and an alignment decoder for implicit duration prediction of the segments.

8.3 Outlook

As outlined in Chapter 1, video understanding systems can be employed for a wide range of different applications and potential use cases, from (semi-)autonomous driving to robot perception and video surveillance. One common underlying aspect that different video understanding tasks share is that they rely on or benefit from a high-quality descriptive video representation. Learning such representations in a supervised setting is prohibitively expensive and does not guarantee optimal features. Therefore, we proposed several self-supervised video representation methods in this thesis, all based on RGB-video data only and trained on curated datasets. In reality, large scale unlabelled data in the wild is much cheaper to obtain than labelled data but uncurated. Therefore, extending self-supervised methods to uncurated data is an important future research direction. Furthermore, RGB videos are often accompanied by other modalities, *e.g.* audio or text. Different modalities provide strong supervision signals to one another and provide the opportunity to develop even more powerful representation learning methods. In the following, we will first discuss the limitations of the methods in this thesis and then discuss some potential future research directions.

8.3.1 Limitations and Future Work

Limited Evaluation. In Chapters 4, 5 and 6, we developed several methods for self-supervised video representation learning, which demonstrate strong empirical performance. However, their evaluation is limited to only a few selected downstream tasks, as is the default evaluation of video representations in the literature. In an attempt to diversify the standard evaluation protocol – which typically only considers action classification downstream tasks – we extend the evaluation with an action segmentation downstream task in Chapter 6. Nevertheless, the video understanding tasks we encounter in practical applications are much more diverse and current evaluation protocols do not assess the applicability of video representations to the wide range of potential downstream tasks. However, this is a more general limitation of the field and applies to the video representation learning literature in general.

Combining BFP with Instance Recognition. In Chapter 4, we developed a bidirectional feature prediction task in a contrastive learning framework, which relies on temporal hard negatives to learn

rich representations. Meanwhile, modern contrastive learning methods are largely based on the concept on instance recognition. These two approaches are not mutually exclusive; in fact, *Dave et al.* (2022) show that by combining an instance recognition contrastive loss with temporal contrastive losses, which include temporal hard negatives, even more powerful representations can be learned. Analogously, future work should investigate the benefit that can be obtained by combining BFP with an instance recognition based loss.

Learning to Rank with RINCE. In Chapter 5, we propose a ranking InfoNCE formulation that allows for a ranked definition of positive pairs and aims to learn a model that mirrors this ranked definition of positives in terms of their similarities in the embedding space. While we mainly apply and evaluate our framework for image and video representation learning, it naturally fits the learning to rank objective as well. In fact, our analysis in Section 5.3.6 and Figure 5.7 suggests that RINCE does indeed learn to rank samples. While it is out of the scope of this thesis, it would be interesting for future work to investigate the potential of our RINCE loss for tasks related to learning to rank.

Temporal Hard Negatives for LSFD. Our BFP method in Chapter 4 designs a set of temporal hard negatives to encourage the model to explore the temporal structure of videos more thoroughly. In contrast, the non-stationary contrastive loss used for our LSFD method in Chapter 6 utilizes only negative samples from different random videos. To bridge the gap between these two approaches for learning temporal features in videos, future work should devise an analogous set of temporal hard negatives for the non-stationary features of LSFD. For example, the positive sample for the non-stationary features of a long view are the aggregated non-stationary features of several short views in correct order; a temporal hard negative could be obtained by randomly permuting or reversing the order of the short views. While we find the Sum aggregation to be superior in Section 6.3.3, temporal hard negatives require an aggregation function that is not invariant to permutations, *e.g.* the other aggregation functions considered in the ablation study, Linear, MLP, or GRU. However, one could argue that such temporal hard negatives should not be considered *true* negatives as they are still highly related to the original sequence. One potential solution to this would be to rank these temporal hard negatives using our RINCE framework from Chapter 5. Here, the non-stationary features of the correctly ordered sequences would be considered higher ranked than those of the incorrectly ordered ones.

Extension to Non-contrastive Learning. The methods in Chapters 4, 5 and 6 were developed and applied in a contrastive learning setting, although some ideas extend to more general concepts. More recently, some methods have been developed for the instance recognition task that omit negative pairs and instead only focus on maximizing the similarity of positive pairs; these methods are generally referred to as non-contrastive methods as they do not contrast the positive pair with negatives. Future works should re-iterate the ideas and concepts introduced in this thesis under a non-contrasive learning framework. For example, the losses in Chapter 6 for learning stationary and non-stationary features do not rely on a specific set of negatives and reformulating them in a non-contrastive framework is straightforward. On the other hand, the methods introduced in Chapter 4 and 5 employ hard negatives of some form: Our BFP method incorporates a set of hard temporal negatives by swapping the order of future and past clips, while RINCE uses lower ranked positives as hard negatives for higher ranked positives to ensure the ranking property in the embedding space. Extending these

methods to a non-contrastive setting is therefore more delicate; at an abstract level, future research can investigate how to leverage unobserved past frames in a non-contrastive video representation learning framework or how to properly handle positive pairs of different similarity.

Improving Frames-to-Segments Alignment in UVAST. Our proposed Transformer-based model for action segmentation in Chapter 7 performs particularly well for transcript prediction as is evident by the high Edit scores our model achieves across all datasets and under different levels of supervision. Nevertheless, our method struggles to localize actions accurately on some of the datasets and future research can target this area for further improvements. While our alignment decoder works well on some of the datasets, others require the assistance of inference algorithms such as Viterbi decoding, which is computationally more expensive. Future work can investigate new methods for faster inference and better alignment and duration prediction.

8.3.2 Large Scale Uncurated and Multi-Modal Video Representation Learning

Working with Uncurated Data. The experiments in this thesis were carried out on curated datasets, such as Kinetics-400 (*Kay et al.*, 2017). In a self-supervised video representation learning setting, we do not rely on the annotated class labels of the datasets to obtain a supervision signal and instead construct a learning signal form the unlabelled data itself. The common practice, however, is to train models on large-scale labelled datasets and simply drop the labels during self-supervised pretraining, which we follow in Chapters 4, 5 and 6 as well. Nevertheless, a supervision bias remains as the original datasets are curated, showing diverse and balanced actions and, importantly, the video clips are pre-segmented to show actions of interest. The true value of self-supervised representation learning lies in its ability to learn from large amounts of unlabelled data, of which massive quantities are available online. Although, the image and video representation learning community is currently mostly focused on training self-supervised methods on curated data, like ImageNet (*Deng et al.*, 2009) or Kinetics (*Kay et al.*, 2017), extending these frameworks to uncurated datasets is of high practical importance. For example, *Goyal et al.* (2021) demonstrate the effectiveness of training the method of (*Caron et al.*, 2020) on large scale uncurated data from the Internet, while *Tian et al.* (2021a) develop a method for applying self-supervised methods on uncurated data more effectively, specifically by addressing the different distribution of uncurated data in comparison to curated data.

Training Big Models. The self-supervised video representation learning methods developed in this thesis were trained and evaluated using relatively shallow architectures such as a 3D-Resnet18. In comparison, the state-of-the-art supervised methods typically utilize significantly deeper architectures that can learn more powerful representations. In Chapter 6 we have seen that self-supervised video representation learning has the potential to outperform supervised pretraining, which we further validated in Chapter 7. However, these findings are restricted to smaller network architectures and the question whether they transfer to bigger architectures remains open. The training of such deep models require a substantial amount of computational resources and are out of the scope of this thesis. One advantage of supervised learning here is that it provides a very strong learning signal to the model, enabling it to learn useful representations with comparatively shorter training times. In contrast, self-supervised learning tends to be more computationally demanding and often requires longer training times as the learning signal is weaker. However, a major disadvantage of supervised

learning is that it requires large scale labelled datasets, which can be prohibitively expensive and time consuming to annotate. Here lies one of the considerable benefits of self-supervised learning as it does not require any labelled data and therefore can be easier scaled up to massive (unlabelled) datasets. Furthermore, the transferability of supervised learned representations largely depends on the labels of the pretraining dataset and how well the pretraining and target tasks are aligned. On the other hand, self-supervised learning allows us to put an explicit focus on a specific type of features by designing a pretext task accordingly. For these reasons, learning very deep models in a self-supervised fashion is an interesting research direction and can be neatly combined with the training on large scale uncurated data mentioned above.

Multi-modal Video Representation Learning. The self-supervised video representation learning methods developed in this thesis consider RGB-video input only, although in practice video data is often accompanied by other modalities such as audio or text. Multi-modal video data naturally provides the opportunity to construct even more powerful self-supervised representation learning tasks, which provide a very strong supervision signal. For example, given one modality, the model needs to extract the underlying shared semantics in order to reason about the other modality. In fact, multi-modal representation learning is already a very active and popular research area. Furthermore, it can naturally be combined with large scale uncurated datasets. A very prominent recent example is the CLIP method (*Radford et al.*, 2021), which trains a model on a huge collection of image-text pairs in a contrastive learning framework to match each image to its corresponding text and achieves impressive performance on various tasks and datasets. Similar directions for multi-modal video representation learning thus seem promising.

Bibliography

Agrawal, Pulkit; Carreira, Joao, and Malik, Jitendra. Learning to see by moving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2015. (Cited on page 13.)

Arnab, Anurag; Dehghani, Mostafa; Heigold, Georg; Sun, Chen; Lučić, Mario, and Schmid, Cordelia. ViViT: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. (Cited on page 24.)

Asano, Yuki M.; Rupprecht, Christian, and Vedaldi, Andrea. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations*, 2020. (Cited on page 49.)

Azizi, Shekoofeh; Mustafa, Basil; Ryan, Fiona; Beaver, Zachary; Freyberg, Jan; Deaton, Jonathan; Loh, Aaron; Karthikesalingam, Alan; Kornblith, Simon; Chen, Ting; Natarajan, Vivek, and Norouzi, Mohammad. Big self-supervised models advance medical image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. (Cited on page 18.)

Ba, Jimmy Lei; Kiros, Jamie Ryan, and Hinton, Geoffrey E. Layer normalization. *arXiv*, abs/1607.06450, 2016. (Cited on page 43.)

Bachman, Philip; Hjelm, R Devon, and Buchwalter, William. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, 2019. (Cited on page 15.)

Bai, Yutong; Fan, Haoqi; Misra, Ishan; Venkatesh, Ganesh; Lu, Yongyi; Zhou, Yuyin; Yu, Qihang; Chandra, Vikas, and Yuille, Alan L. Can temporal information help with contrastive self-supervised learning? *arXiv*, abs/2011.13046, 2020. (Cited on pages 5 and 19.)

Behrmann, Nadine; Fayyaz, Mohsen; Gall, Juergen, and Noroozi, Mehdi. Long short view feature decomposition via contrastive video representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021a. (Cited on pages 8 and 85.)

Behrmann, Nadine; Gall, Juergen, and Noroozi, Mehdi. Unsupervised video representation learning by bidirectional feature prediction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021b. (Cited on pages 7 and 47.)

Behrmann, Nadine; Golestaneh, S. Alireza; Kolter, Zico; Gall, Juergen, and Noroozi, Mehdi. Unified fully and timestamp supervised temporal action segmentation via sequence to sequence translation. In *Proceedings of the European Conference on Computer Vision*, 2022. (Cited on pages 9 and 103.)

Belghazi, Mohamed Ishmael; Baratin, Aristide; Rajeshwar, Sai; Ozair, Sherjil; Bengio, Yoshua; Courville, Aaron, and Hjelm, Devon. Mutual information neural estimation. In *Proceedings of the International Conference on Machine Learning*, 2018. (Cited on page 15.)

Benaim, Sagie; Ephrat, Ariel; Lang, Oran; Mosseri, Inbar; Freeman, William T.; Rubinstein, Michael; Irani, Michal, and Dekel, Tali. SpeedNet: Learning the speediness in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. (Cited on pages 13, 56, 87, 92 and 97.)

Bengio, Yoshua; Courville, Aaron, and Vincent, Pascal. Representation learning: a review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013. (Cited on pages 29 and 31.)

Bertasius, Gedas; Wang, Heng, and Torresani, Lorenzo. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning*, 2021. (Cited on page 24.)

Bojanowski, Piotr; Lajugie, Rémi; Bach, Francis R.; Laptev, Ivan; Ponce, Jean; Schmid, Cordelia, and Sivic, Josef. Weakly supervised action labeling in videos under ordering constraints. In *Proceedings of the European Conference on Computer Vision*, 2014. (Cited on page 27.)

Brown, Tom; Mann, Benjamin; Ryder, Nick; Subbiah, Melanie; Kaplan, Jared D; Dhariwal, Prafulla; Neelakantan, Arvind; Shyam, Pranav; Sastry, Girish; Askell, Amanda; Agarwal, Sandhini; Herbert-Voss, Ariel; Krueger, Gretchen; Henighan, Tom; Child, Rewon; Ramesh, Aditya; Ziegler, Daniel; Wu, Jeffrey; Winter, Clemens; Hesse, Chris; Chen, Mark; Sigler, Eric; Litwin, Mateusz; Gray, Scott; Chess, Benjamin; Clark, Jack; Berner, Christopher; McCandlish, Sam; Radford, Alec; Sutskever, Ilya, and Amodei, Dario. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020. (Cited on page 2.)

Cakir, Fatih; He, Kun; Xia, Xide; Kulis, Brian, and Sclaroff, Stan. Deep metric learning to rank. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. (Cited on page 74.)

Caron, Mathilde; Misra, Ishan; Mairal, Julien; Goyal, Priya; Bojanowski, Piotr, and Joulin, Armand. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, 2020. (Cited on pages 17, 19 and 128.)

Caron, Mathilde; Touvron, Hugo; Misra, Ishan; Jégou, Hervé; Mairal, Julien; Bojanowski, Piotr, and Joulin, Armand. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. (Cited on page 22.)

Carreira, João and Zisserman, Andrew. Quo vadis, action recognition? A new model and the kinetics dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017. (Cited on pages 11, 23, 98, 112 and 116.)

Cer, Daniel; Diab, Mona; Agirre, Eneko; Lopez-Gazpio, Inigo, and Specia, Lucia. SemEval-2017 task 1: Semantic textual similarity multilingual and cross-lingual focused evaluation. *arXiv*, abs/1708.00055, 2017. (Cited on page 79.)

Chaitanya, Krishna; Erdil, Ertunc; Karani, Neerav, and Konukoglu, Ender. Contrastive learning of global and local features for medical image segmentation with limited annotations. In *Advances in Neural Information Processing Systems*, 2020. (Cited on page 18.)

Chang, Chien-Yi; Huang, De-An; Sui, Yanan; Fei-Fei, Li, and Niebles, Juan Carlos. D3TW: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. (Cited on page 27.)

Chang, Xiaobin; Tung, Frederick, and Mori, Greg. Learning discriminative prototypes with dynamic time warping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. (Cited on page 27.)

Chen, Min-Hung; Li, Baopu; Bao, Yingze; AlRegib, Ghassan, and Kira, Zsolt. Action segmentation with joint self-supervised temporal domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020a. (Cited on pages 111 and 113.)

Chen, Peihao; Huang, Deng; He, Dongliang; Long, Xiang; Zeng, Runhao; Wen, Shilei; Tan, Mingkui, and Gan, Chuang. RSPNet: Relative speed perception for unsupervised video representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021a. (Cited on pages 56, 81 and 92.)

Chen, Shixing; Nie, Xiaohan; Fan, David; Zhang, Dongqing; Bhat, Vimal, and Hamid, Raffay. Shot contrastive self-supervised learning for scene boundary detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021b. (Cited on page 18.)

Chen, Ting; Kornblith, Simon; Norouzi, Mohammad, and Hinton, Geoffrey. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference on Machine Learning*, 2020b. (Cited on pages 2, 14, 16, 18, 19, 33, 49, 52, 66, 67, 81 and 90.)

Chen, Ting; Kornblith, Simon; Swersky, Kevin; Norouzi, Mohammad, and Hinton, Geoffrey E. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems*, 2020c. (Cited on page 18.)

Chen, Xinlei and He, Kaiming. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. (Cited on pages 21 and 22.)

Chen, Xinlei; Fan, Haoqi; Girshick, Ross, and He, Kaiming. Improved baselines with momentum contrastive learning. *arXiv*, abs/2003.04297, 2020d. (Cited on page 73.)

Cho, Kyunghyun; van Merriënboer, Bart; Bahdanau, Dzmitry, and Bengio, Yoshua. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014. (Cited on page 42.)

Dave, Ishan; Gupta, Rohit; Rizve, Mamshad Nayeem, and Shah, Mubarak. TCLR: Temporal contrastive learning for video representation. *Computer Vision and Image Understanding*, 2022. (Cited on pages 8, 20, 56, 57, 65, 81, 92 and 127.)

Deng, Jia; Dong, Wei; Socher, Richard; Li, Li-Jia; Li, Kai, and Fei-Fei, Li. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009. (Cited on pages 23, 74 and 128.)

Deselaers, Thomas and Ferrari, Vittorio. Visual and semantic similarity in ImageNet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2011. (Cited on pages 8, 65 and 74.)

Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton, and Toutanova, Kristina. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019. (Cited on page 2.)

Ding, Li and Xu, Chenliang. Weakly-supervised action segmentation with iterative soft boundary assignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on page 27.)

Donahue, Jeff; Hendricks, Lisa Anne; Rohrbach, Marcus; Venugopalan, Subhashini; Guadarrama, Sergio; Saenko, Kate, and Darrell, Trevor. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015. (Cited on page 22.)

Dosovitskiy, Alexey; Fischer, Philipp; Springenberg, Jost Tobias; Riedmiller, Martin, and Brox, Thomas. Discriminative unsupervised feature learning with exemplar convolutional neural networks. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. (Cited on page 16.)

Dosovitskiy, Alexey; Beyer, Lucas; Kolesnikov, Alexander; Weissenborn, Dirk; Zhai, Xiaohua; Unterthiner, Thomas; Dehghani, Mostafa; Minderer, Matthias; Heigold, Georg; Gelly, Sylvain; Uszkoreit, Jakob, and Houlsby, Neil. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. (Cited on pages 23 and 42.)

Dwibedi, Debidatta; Aytar, Yusuf; Tompson, Jonathan; Sermanet, Pierre, and Zisserman, Andrew. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. (Cited on page 17.)

Epstein, Dave; Chen, Boyuan, and Vondrick, Carl. Oops! predicting unintentional action in video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. (Cited on pages 13 and 14.)

Ericsson, Linus; Gouk, Henry, and Hospedales, Timothy M. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. (Cited on page 18.)

Farha, Yazan Abu and Gall, Juergen. MS-TCN: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. (Cited on pages 26, 38, 98, 111, 114 and 115.)

Fathi, Alireza; Ren, Xiaofeng, and Rehg, James M. Learning to recognize objects in egocentric activities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2011. (Cited on pages 27, 28, 106 and 111.)

Feichtenhofer, Christoph; Pinz, Axel, and Zisserman, Andrew. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 23.)

Feichtenhofer, Christoph; Fan, Haoqi; Malik, Jitendra, and He, Kaiming. SlowFast networks for video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. (Cited on pages 11 and 23.)

Feichtenhofer, Christoph; Fan, Haoqi; Xiong, Bo; Girshick, Ross, and He, Kaiming. A large-scale study on unsupervised spatiotemporal representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. (Cited on pages 8, 19, 34 and 65.)

Fernando, Basura; Bilen, Hakan; Gavves, Efstratios, and Gould, Stephen. Self-supervised video representation learning with odd-one-out networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017. (Cited on pages 12 and 56.)

Fouhey, David F.; Kuo, Weicheng; Efros, Alexei A., and Malik, Jitendra. From lifestyle VLOGs to everyday interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on page 49.)

Gammulle, Harshala; Denman, Simon; Sridharan, Sridha, and Fookes, Clinton. Fine-grained action segmentation using the semi-supervised action gan. *Pattern Recognition*, 2020. (Cited on page 113.)

Gao, Shang-Hua; Han, Qi; Li, Zhong-Yu; Peng, Pai; Wang, Liang, and Cheng, Ming-Ming. Global2local: Efficient structure search for video action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. (Cited on page 113.)

Ge, Weifeng. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision*, 2018. (Cited on page 74.)

Gidaris, Spyros; Singh, Praveer, and Komodakis, Nikos. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018. (Cited on page 12.)

Goyal, Priya; Caron, Mathilde; Lefaudeux, Benjamin; Xu, Min; Wang, Pengchao; Pai, Vivek; Singh, Mannat; Liptchinsky, Vitaliy; Misra, Ishan; Joulin, Armand, and Bojanowski, Piotr. Self-supervised pretraining of visual features in the wild. *arXiv*, abs/2103.01988, 2021. (Cited on page 128.)

Goyal, Raghav; Kahou, Samira Ebrahimi; Michalski, Vincent; Materzynska, Joanna; Westphal, Susanne; Kim, Heuna; Haenel, Valentin; Fruend, Ingo; Yianilos, Peter; Mueller-Freitag, Moritz; Hoppe, Florian; Thurau, Christian; Bax, Ingo, and Memisevic, Roland. The "Something Something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2017. (Cited on pages 3 and 12.)

Grill, Jean-Bastien; Strub, Florian; Altché, Florent; Tallec, Corentin; Richemond, Pierre; Buchatskaya, Elena; Doersch, Carl; Avila Pires, Bernardo; Guo, Zhaohan; Gheshlaghi Azar, Mohammad; Piot, Bilal; Kavukcuoglu, Koray; Munos, Rémi, and Valko, Michal. Bootstrap your own

latent - a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, 2020. (Cited on pages 19, 21, 22 and 49.)

Han, Tengda; Xie, Weidi, and Zisserman, Andrew. Video representation learning by dense predictive coding. In *Proceedings of the IEEE/CVF International Conference on Computer VisionWorkshop*, 2019. (Cited on pages 14, 17, 20, 41, 52, 53, 54, 55, 56, 57 and 92.)

Han, Tengda; Xie, Weidi, and Zisserman, Andrew. Self-supervised co-training for video representation learning. In *Advances in Neural Information Processing Systems*, 2020a. (Cited on pages 17, 20, 56, 69, 92, 97 and 98.)

Han, Tengda; Xie, Weidi, and Zisserman, Andrew. Memory-augmented dense predictive coding for video representation learning. In *Proceedings of the European Conference on Computer Vision*, 2020b. (Cited on pages 14, 17, 20, 56, 92 and 97.)

Hara, Kensho; Kataoka, Hirokatsu, and Satoh, Yutaka. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on pages 11, 23, 40, 41, 82, 91 and 116.)

He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on pages 23 and 40.)

He, Kaiming; Girshick, Ross, and Dollar, Piotr. Rethinking ImageNet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. (Cited on page 91.)

He, Kaiming; Fan, Haoqi; Wu, Yuxin; Xie, Saining, and Girshick, Ross. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. (Cited on pages 2, 17, 19, 81, 82, 93 and 94.)

Hendrycks, Dan; Mazeika, Mantas; Kadavath, Saurav, and Song, Dawn. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems*, 2019. (Cited on page 76.)

Hjelm, R Devon; Fedorov, Alex; Lavoie-Marchildon, Samuel; Grewal, Karan; Bachman, Phil; Trischler, Adam, and Bengio, Yoshua. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019. (Cited on page 15.)

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural Computation*, 1997. (Cited on pages 22 and 42.)

Hoffmann, David; Behrmann, Nadine; Gall, Juergen; Brox, Thomas, and Noroozi, Mehdi. Ranking info noise contrastive estimation: Boosting contrastive learning via ranked positives. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. (Cited on pages 8 and 63.)

Hornik, Kurt; Stinchcombe, Maxwell, and White, Halbert. Multilayer feedforward networks are universal approximators. *Neural Networks*, 1989. (Cited on page 32.)

Huang, De-An; Fei-Fei, Li, and Niebles, Juan Carlos. Connectionist temporal modeling for weakly supervised action labeling. In *Proceedings of the European Conference on Computer Vision*, 2016. (Cited on page 27.)

Huang, Deng; Wu, Wenhao; Hu, Weiwen; Liu, Xu; He, Dongliang; Wu, Zhihua; Wu, Xiangmiao; Tan, Mingkui, and Ding, Errui. ASCNet: Self-supervised video representation learning with appearance-speed consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. (Cited on pages 56, 81 and 92.)

Huang, Yifei; Sugano, Yusuke, and Sato, Yoichi. Improving action segmentation via graph-based temporal reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. (Cited on page 26.)

Huynh, Tri; Kornblith, Simon; Walter, Matthew R; Maire, Michael, and Khademi, Maryam. Boosting contrastive self-supervised learning with false negative cancellation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022. (Cited on pages 17 and 18.)

ICCV, . Paper visualizations, 2021. URL `https://iccv2021.thecvf.com/papers-visualizations?filter=authors`. (Cited on page 3.)

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*, 2015. (Cited on pages 22 and 41.)

Ishikawa, Yuchi; Kasai, Seito; Aoki, Yoshimitsu, and Kataoka, Hirokatsu. Alleviating oversegmentation errors by detecting action boundaries. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021. (Cited on pages 26, 111 and 113.)

Isola, Phillip; Zoran, Daniel; Krishnan, Dilip, and Adelson, Edward H. Learning visual groups from co-occurrences in space and time. In *International Conference on Learning RepresentationsWorkshop*, 2016. (Cited on page 13.)

Jayaraman, Dinesh and Grauman, Kristen. Learning image representations tied to ego-motion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2015. (Cited on page 13.)

Jayaraman, Dinesh and Grauman, Kristen. Slow and steady feature analysis: Higher order temporal coherence in video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 13.)

Jenni, Simon and Jin, Hailin. Time-equivariant contrastive video representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. (Cited on pages 19, 56, 57, 81, 82, 92 and 95.)

Jenni, Simon; Meishvili, Givi, and Favaro, Paolo. Video representation learning by recognizing temporal transformations. In *Proceedings of the European Conference on Computer Vision*, 2020. (Cited on pages 14, 56, 81, 87, 92, 95 and 97.)

Jing, Longlong and Tian, Yingli. Self-supervised spatiotemporal feature learning via video rotation prediction. *arXiv*, abs/1811.11387, 2018. (Cited on pages 12, 56, 81 and 92.)

Jing, Longlong and Tian, Yingli. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. (Cited on page 12.)

Karaman, Svebor; Seidenari, Lorenzo, and Bimbo, Alberto Del. Fast saliency based pooling of fisher encoded dense trajectories. In *Proceedings of the European Conference on Computer VisionWorkshops*, 2014. (Cited on page 25.)

Karpathy, Andrej; Toderici, George; Shetty, Sanketh; Leung, Thomas; Sukthankar, Rahul, and Fei-Fei, Li. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2014. (Cited on pages 4 and 22.)

Kay, Will; Carreira, Joao; Simonyan, Karen; Zhang, Brian; Hillier, Chloe; Vijayanarasimhan, Sudheendra; Viola, Fabio; Green, Tim; Back, Trevor; Natsev, Paul; Suleyman, Mustafa, and Zisserman, Andrew. The kinetics human action video dataset. *arXiv*, abs/1705.06950, 2017. (Cited on pages 2, 12, 24, 25, 57, 82, 87, 88, 93 and 128.)

Khosla, Prannay; Teterwak, Piotr; Wang, Chen; Sarna, Aaron; Tian, Yonglong; Isola, Phillip; Maschinot, Aaron; Liu, Ce, and Krishnan, Dilip. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, 2020. (Cited on pages 7, 17, 18, 33, 65, 66, 67, 69, 73 and 74.)

Kim, Dahun; Cho, Donghyeon, and Kweon, In So. Self-supervised video representation learning with space-time cubic puzzles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019. (Cited on pages 12, 56, 81 and 92.)

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. (Cited on pages 54, 82, 93 and 112.)

Kolesnikov, Alexander; Zhai, Xiaohua, and Beyer, Lucas. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. (Cited on pages 34 and 37.)

Krizhevsky, Alex. One weird trick for parallelizing convolutional neural networks. *arXiv*, abs/1404.5997, 2014. (Cited on page 73.)

Krizhevsky, Alex; Hinton, Geoffrey, and others, . Learning multiple layers of features from tiny images. *Tech Report*, 2009. (Cited on page 74.)

Kuang, Haofei; Zhu, Yi; Zhang, Zhi; Li, Xinyu; Tighe, Joseph; Schwertfeger, Sören; Stachniss, Cyrill, and Li, Mu. Video contrastive learning with global context. In *Proceedings of the IEEE/CVF International Conference on Computer VisionWorkshops*, 2021. (Cited on pages 19 and 20.)

Kuehne, Hilde; Jhuang, Hueihan; Garrote, Estíbaliz; Poggio, Tomaso, and Serre, Thomas. HMDB: A large video database for human motion recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2011. (Cited on pages 24, 54, 82, 88 and 94.)

Kuehne, Hilde; Arslan, Ali B., and Serre, Thomas. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2014. (Cited on pages 27, 28, 88, 94, 106 and 111.)

Kuehne, Hilde; Gall, Juergen, and Serre, Thomas. An end-to-end generative framework for video segmentation and recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2016. (Cited on page 25.)

Kuehne, Hilde; Richard, Alexander, and Gall, Juergen. A hybrid RNN-HMM approach for weakly supervised temporal action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. (Cited on page 106.)

Lai, Zihang and Xie, Weidi. Self-supervised learning for video correspondence flow. In *British Machine Vision Conference*, 2019. (Cited on page 13.)

Laptev, Ivan and Lindeberg, Tony. Space-time interest points. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2003. (Cited on page 30.)

Larsson, Gustav; Maire, Michael, and Shakhnarovich, Gregory. Learning representations for automatic colorization. In *Proceedings of the European Conference on Computer Vision*, 2016. (Cited on page 14.)

Le, Ya and Yang, Xuan. Tiny ImageNet visual recognition challenge, 2015. (Cited on page 74.)

Lea, Colin; Reiter, Austin; Vidal, René, and Hager, Gregory D. Segmental spatiotemporal CNNs for fine-grained action segmentation. In *Proceedings of the European Conference on Computer Vision*, 2016a. (Cited on page 25.)

Lea, Colin; Vidal, René, and Hager, Gregory D. Learning convolutional action primitives for fine-grained action recognition. In *International Conference on Robotics and Automation*, 2016b. (Cited on page 39.)

Lea, Colin; Flynn, Michael D.; Vidal, René; Reiter, Austin, and Hager, Gregory D. Temporal convolutional networks for action segmentation and detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017. (Cited on pages 25, 39, 99 and 111.)

Lee, Doyup and Cheon, Yeongjae. Soft labeling affects out-of-distribution detection of deep neural networks. *arXiv*, abs/2007.03212, 2020. (Cited on pages 74 and 77.)

Lee, Hsin-Ying; Huang, Jia-Bin; Singh, Maneesh, and Yang, Ming-Hsuan. Unsupervised representation learning by sorting sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2017. (Cited on pages 12, 56 and 92.)

Lee, Kimin; Lee, Kibok; Lee, Honglak, and Shin, Jinwoo. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, 2018. (Cited on pages 77 and 78.)

Lei, Peng and Todorovic, Sinisa. Temporal deformable residual networks for action segmentation in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on pages 26 and 113.)

Lenc, Karel and Vedaldi, Andrea. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015. (Cited on page 95.)

Li, Jun; Lei, Peng, and Todorovic, Sinisa. Weakly supervised energy-based learning for action segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. (Cited on page 27.)

Li, Junnan; Zhou, Pan; Xiong, Caiming, and Hoi, Steven. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*, 2021a. (Cited on page 17.)

Li, Rui; Zhang, Yiheng; Qiu, Zhaofan; Yao, Ting; Liu, Dong, and Mei, Tao. Motion-focused contrastive learning of video representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021b. (Cited on page 92.)

Li, Shi-Jie; AbuFarha, Yazan; Liu, Yun; Cheng, Ming-Ming, and Gall, Juergen. MS-TCN++: Multi-stage temporal convolutional network for action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. (Cited on pages 26, 98, 111, 113 and 114.)

Li, Yingwei; Li, Yi, and Vasconcelos, Nuno. RESOUND: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision*, 2018. (Cited on pages 3, 12, 38, 49, 82, 83, 86, 87, 91, 95 and 98.)

Li, Zhe; Abu Farha, Yazan, and Gall, Juergen. Temporal action segmentation from timestamp supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021c. (Cited on pages 27, 114 and 115.)

Liang, Shiyu; Li, Yixuan, and Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018. (Cited on pages 77 and 78.)

Liu, Yinhan; Ott, Myle; Goyal, Naman; Du, Jingfei; Joshi, Mandar; Chen, Danqi; Levy, Omer; Lewis, Mike; Zettlemoyer, Luke, and Stoyanov, Veselin. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv*, abs/1907.11692, 2019. (Cited on pages 66, 74 and 78.)

Liu, Ze; Lin, Yutong; Cao, Yue; Hu, Han; Wei, Yixuan; Zhang, Zheng; Lin, Stephen, and Guo, Baining. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021a. (Cited on pages 24 and 42.)

Liu, Ze; Ning, Jia; Cao, Yue; Wei, Yixuan; Zhang, Zheng; Lin, Stephen, and Hu, Han. Video swin transformer. *arXiv*, abs/2106.13230, 2021b. (Cited on page 24.)

Locatello, Francesco; Bauer, Stefan; Lucic, Mario; Raetsch, Gunnar; Gelly, Sylvain; Schölkopf, Bernhard, and Bachem, Olivier. Challenging common assumptions in the unsupervised learning of disentangled representations. In *Proceedings of the International Conference on Machine Learning*, 2019. (Cited on page 31.)

Lotter, William; Kreiman, Gabriel, and Cox, David D. Deep predictive coding networks for video prediction and unsupervised learning. In *International Conference on Learning Representations*, 2017. (Cited on page 14.)

Luo, Dezhao; Liu, Chang; Zhou, Yu; Yang, Dongbao; Ma, Can; Ye, Qixiang, and Wang, Weiping. Video cloze procedure for self-supervised spatio-temporal learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. (Cited on pages 13 and 97.)

Mathieu, Michaël; Couprie, Camille, and LeCun, Yann. Deep multi-scale video prediction beyond mean square error. In *International Conference on Learning Representations*, 2016. (Cited on pages 14 and 49.)

Miech, Antoine; Alayrac, Jean-Baptiste; Smaira, Lucas; Laptev, Ivan; Sivic, Josef, and Zisserman, Andrew. End-to-end learning of visual representations from uncurated instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. (Cited on pages 67 and 69.)

Misra, Ishan and van der Maaten, Laurens. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. (Cited on pages 16 and 52.)

Misra, Ishan; Zitnick, C. Lawrence, and Hebert, Martial. Shuffle and learn: Unsupervised learning using temporal order verification. In *Proceedings of the European Conference on Computer Vision*, 2016. (Cited on pages 12, 56 and 92.)

Neill, James O' and Bollegala, Danushka. Semantically-conditioned negative samples for efficient contrastive learning. *arXiv*, abs/2102.06603, 2021. (Cited on page 18.)

Noroozi, Mehdi and Favaro, Paolo. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proceedings of the European Conference on Computer Vision*, 2016. (Cited on pages 12, 34, 36 and 53.)

Noroozi, Mehdi; Pirsiavash, Hamed, and Favaro, Paolo. Representation learning by learning to count. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2017. (Cited on pages 16 and 96.)

Pan, Tian; Song, Yibing; Yang, Tianyu; Jiang, Wenhao, and Liu, Wei. VideoMoCo: Contrastive video representation learning with temporally adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. (Cited on page 19.)

Pathak, Deepak; Krähenbühl, Philipp; Donahue, Jeff; Darrell, Trevor, and Efros, Alexei A. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 14.)

Patrick, Mandela; Asano, Yuki M.; Kuznetsova, Polina; Fong, Ruth; Henriques, João F.; Zweig, Geoffrey, and Vedaldi, Andrea. On compositions of transformations in contrastive self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. (Cited on pages 17, 21 and 95.)

Pedregosa, Fabian; Varoquaux, Gaël; Gramfort, Alexandre; Michel, Vincent; Thirion, Bertrand; Grisel, Olivier; Blondel, Mathieu; Prettenhofer, Peter; Weiss, Ron; Dubourg, Vincent; Vanderplas, Jake; Passos, Alexandre; Cournapeau, David; Brucher, Matthieu; Perrot, Matthieu, and Édouard Duchesnay, . Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011. (Cited on page 78.)

Piergiovanni, AJ; Angelova, Anelia, and Ryoo, Michael S. Evolving losses for unsupervised video representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. (Cited on pages 49 and 95.)

Poole, Ben; Ozair, Sherjil; van den Oord, Aaron; Alemi, Alex, and Tucker, George. On variational bounds of mutual information. In *Proceedings of the International Conference on Machine Learning*, 2019. (Cited on pages 14, 34 and 53.)

Purushwalkam, Senthil and Gupta, Abhinav. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. In *Advances in Neural Information Processing Systems*, 2020. (Cited on pages 16 and 35.)

Qian, Rui; Li, Yuxi; Liu, Huabin; See, John; Ding, Shuangrui; Liu, Xian; Li, Dian, and Lin, Weiyao. Enhancing self-supervised video representation learning via multi-level feature optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021a. (Cited on pages 56, 81 and 92.)

Qian, Rui; Meng, Tianjian; Gong, Boqing; Yang, Ming-Hsuan; Wang, Huisheng; Belongie, Serge, and Cui, Yin. Spatiotemporal contrastive video representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021b. (Cited on pages 19, 56 and 92.)

Qian, Rui; Lin, Weiyao; See, John, and Li, Dian. Controllable augmentations for video representation learning. *arXiv*, abs/2203.16632, 2022. (Cited on page 20.)

Qiu, Zhaofan; Yao, Ting, and Mei, Tao. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2017. (Cited on page 23.)

Radford, Alec; Kim, Jong Wook; Hallacy, Chris; Ramesh, Aditya; Goh, Gabriel; Agarwal, Sandhini; Sastry, Girish; Askell, Amanda; Mishkin, Pamela; Clark, Jack; Krueger, Gretchen, and Sutskever, Ilya. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning*, 2021. (Cited on pages 2, 17, 21 and 129.)

Recasens, Adrià; Luc, Pauline; Alayrac, Jean-Baptiste; Wang, Luyu; Strub, Florian; Tallec, Corentin; Malinowski, Mateusz; Pătrăucean, Viorica; Altché, Florent; Valko, Michal; Grill, Jean-Bastien; van den Oord, Aäron, and Zisserman, Andrew. Broaden your views for self-supervised video

learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. (Cited on pages 19, 22, 56 and 92.)

Reed, Colorado J; Metzger, Sean; Srinivas, Aravind; Darrell, Trevor, and Keutzer, Kurt. Self-Augment: Automatic augmentation policies for self-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. (Cited on page 16.)

Reimers, Nils and Gurevych, Iryna. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019. (Cited on page 78.)

Richard, Alexander and Gall, Juergen. Temporal action detection using a statistical language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 25.)

Richard, Alexander; Kuehne, Hilde, and Gall, Juergen. Weakly supervised action learning with RNN based fine-to-coarse modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017. (Cited on page 48.)

Richard, Alexander; Kuehne, Hilde; Iqbal, Ahsan, and Gall, Juergen. NeuralNetwork-Viterbi: A framework for weakly supervised video learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on pages 27 and 106.)

Richemond, Pierre H.; Grill, Jean-Bastien; Altché, Florent; Tallec, Corentin; Strub, Florian; Brock, Andrew; Smith, Samuel; De, Soham; Pascanu, Razvan; Piot, Bilal, and Valko, Michal. BYOL works even without batch statistics. In *Advances in Neural Information Processing SystemsWorkshops*, 2020. (Cited on page 22.)

Robinson, Joshua David; Chuang, Ching-Yao; Sra, Suvrit, and Jegelka, Stefanie. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*, 2021. (Cited on page 17.)

Rohrbach, Marcus; Amin, Sikandar; Andriluka, Mykhaylo, and Schiele, Bernt. A database for fine grained activity detection of cooking activities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2012. (Cited on page 25.)

Romijnders, Rob; Mahendran, Aravindh; Tschannen, Michael; Djolonga, Josip; Ritter, Marvin; Houlsby, Neil, and Lucic, Mario. Representation learning from videos in-the-wild: An object-centric approach. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021. (Cited on page 18.)

Sastry, Chandramouli Shama and Oore, Sageev. Detecting out-of-distribution examples with gram matrices. In *Proceedings of the International Conference on Machine Learning*, 2020. (Cited on page 77.)

Schindler, Konrad and van Gool, Luc. Action snippets: How many frames does human action recognition require? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2008. (Cited on page 3.)

Shao, Dian; Zhao, Yue; Dai, Bo, and Lin, Dahua. FineGym: A hierarchical video dataset for fine-grained action understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. (Cited on pages 3, 4 and 12.)

Simonyan, Karen and Zisserman, Andrew. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, 2014. (Cited on page 23.)

Singhania, Dipika; Rahaman, Rahul, and Yao, Angela. Coarse to fine multi-resolution temporal convolutional network. *arXiv*, abs/2105.10859, 2021. (Cited on pages 111 and 113.)

Sohn, Kihyuk. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, 2016. (Cited on page 15.)

Soomro, Khurram; Zamir, Amir Roshan, and Shah, Mubarak. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv*, abs/1212.0402, 2012. (Cited on pages 24, 25, 54, 82, 88 and 94.)

Souri, Yaser; Abu Farha, Yazan; Despinoy, Fabien; Francesca, Gianpiero, and Gall, Juergen. FIFA: Fast inference approximation for action segmentation. In *German Conference for Pattern Recognition*, 2021a. (Cited on pages 106, 113 and 114.)

Souri, Yaser; Fayyaz, Mohsen; Minciullo, Luca; Francesca, Gianpiero, and Gall, Juergen. Fast weakly supervised action segmentation using mutual consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021b. (Cited on pages 27 and 113.)

Srivastava, Nitish; Mansimov, Elman, and Salakhutdinov, Ruslan. Unsupervised learning of video representations using LSTMs. In *Proceedings of the International Conference on Machine Learning*, 2015. (Cited on page 14.)

statista, . Hours of video uploaded to youtube every minute as of february 2020, 2022. URL `https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/#statisticContainer`. (Cited on page 2.)

Stein, Sebastian and McKenna, Stephen J. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2013. (Cited on pages 27, 28, 106 and 111.)

Sun, Chen; Baradel, Fabien; Murphy, Kevin, and Schmid, Cordelia. Learning video representations using contrastive bidirectional transformer. *arXiv*, abs/1906.05743, 2019. (Cited on page 92.)

Sun, Chen; Nagrani, Arsha; Tian, Yonglong, and Schmid, Cordelia. Composable augmentation encoding for video representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. (Cited on pages 20, 56 and 92.)

Surís, Dídac; Liu, Ruoshi, and Vondrick, Carl. Learning the predictability of the future. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. (Cited on page 14.)

Szegedy, Christian; Liu, Wei; Jia, Yangqing; Sermanet, Pierre; Reed, Scott; Anguelov, Dragomir; Erhan, Dumitru; Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015. (Cited on page 23.)

Szegedy, Christian; Vanhoucke, Vincent; Ioffe, Sergey; Shlens, Jon, and Wojna, Zbigniew. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016. (Cited on page 74.)

Tang, Kevin; Fei-Fei, Li, and Koller, Daphne. Learning latent temporal structure for complex event detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2012. (Cited on page 25.)

Tao, Li; Wang, Xueting, and Yamasaki, Toshihiko. Self-supervised video representation learning using inter-intra contrastive framework. In *Proceedings of the 28th ACM International Conference on Multimedia*, 2020. (Cited on page 20.)

Tian, Yonglong; Krishnan, Dilip, and Isola, Phillip. Contrastive multiview coding. In *Proceedings of the European Conference on Computer Vision*, 2020a. (Cited on pages 16 and 74.)

Tian, Yonglong; Krishnan, Dilip, and Isola, Phillip. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020b. (Cited on page 18.)

Tian, Yonglong; Sun, Chen; Poole, Ben; Krishnan, Dilip; Schmid, Cordelia, and Isola, Phillip. What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems*, 2020c. (Cited on page 18.)

Tian, Yonglong; Hénaff, Olivier J., and van den Oord, Aäron. Divide and contrast: Self-supervised learning from uncurated data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021a. (Cited on page 128.)

Tian, Yuandong; Chen, Xinlei, and Ganguli, Surya. Understanding self-supervised learning dynamics without contrastive pairs. In *Proceedings of the International Conference on Machine Learning*, 2021b. (Cited on page 22.)

Tokmakov, Pavel; Hebert, Martial, and Schmid, Cordelia. Unsupervised learning of video representations via dense trajectory clustering. In *Proceedings of the European Conference on Computer VisionWorkshops*, 2020. (Cited on pages 17, 20, 56, 81, 82 and 92.)

Tran, Du; Bourdev, Lubomir; Fergus, Rob; Torresani, Lorenzo, and Paluri, Manohar. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2015. (Cited on page 23.)

Tran, Du; Wang, Heng; Torresani, Lorenzo; Ray, Jamie; LeCun, Yann, and Paluri, Manohar. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on page 23.)

Tschannen, Michael; Djolonga, Josip; Ritter, Marvin; Mahendran, Aravindh; Houlsby, Neil; Gelly, Sylvain, and Lucic, Mario. Self-supervised learning of video-induced visual invariances. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020a. (Cited on page 80.)

Tschannen, Michael; Djolonga, Josip; Rubenstein, Paul K.; Gelly, Sylvain, and Lucic, Mario. On mutual information maximization for representation learning. In *International Conference on Learning Representations*, 2020b. (Cited on pages 15, 34, 49 and 53.)

Ulyanov, Dmitry; Vedaldi, Andrea, and Lempitsky, Victor. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017. (Cited on pages 44 and 45.)

van den Oord, Aäron; Li, Yazhe, and Vinyals, Oriol. Representation learning with contrastive predictive coding. *arXiv*, abs/1807.03748, 2018. (Cited on pages 14, 15, 17, 20, 34 and 51.)

Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Ł ukasz, and Polosukhin, Illia. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. (Cited on pages 9, 10, 23, 26, 40, 42, 44, 106, 112 and 120.)

Vondrick, Carl; Pirsiavash, Hamed, and Torralba, Antonio. Anticipating visual representations from unlabeled video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016a. (Cited on page 14.)

Vondrick, Carl; Pirsiavash, Hamed, and Torralba, Antonio. Generating videos with scene dynamics. In *Advances in Neural Information Processing Systems*, 2016b. (Cited on page 14.)

Vondrick, Carl; Shrivastava, Abhinav; Fathi, Alireza; Guadarrama, Sergio, and Murphy, Kevin. Tracking emerges by colorizing videos. In *Proceedings of the European Conference on Computer Vision*, 2018. (Cited on page 13.)

Wang, Dong; Hu, Di; Li, Xingjian, and Dou, Dejing. Temporal relational modeling with self-supervision for action segmentation. In *Advances in Neural Information Processing Systems*, 2021. (Cited on page 113.)

Wang, Feng and Liu, Huaping. Understanding the behaviour of contrastive loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. (Cited on pages 68, 69 and 70.)

Wang, Feng; Liu, Huaping; Guo, Di, and Fuchun, Sun. Unsupervised representation learning by invariance propagation. In *Advances in Neural Information Processing Systems*, 2020a. (Cited on page 17.)

Wang, Heng and Schmid, Cordelia. Action recognition with improved trajectories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2013. (Cited on pages 11, 30 and 98.)

Wang, Jiangliu; Jiao, Jianbo; Bao, Linchao; He, Shengfeng; Liu, Yunhui, and Liu, Wei. Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019a. (Cited on page 92.)

Wang, Jiangliu; Jiao, Jianbo, and Liu, Yun-Hui. Self-supervised video representation learning by pace prediction. In *Proceedings of the European Conference on Computer Vision*, 2020b. (Cited on pages 14, 19 and 92.)

Wang, Jue; Bertasius, Gedas; Tran, Du, and Torresani, Lorenzo. Long-short temporal contrastive learning of video transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. (Cited on page 19.)

Wang, Limin; Xiong, Yuanjun; Wang, Zhe; Qiao, Yu; Lin, Dahua; Tang, Xiaoou, and Gool, Luc Van. Temporal segment networks for action recognition in videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019b. (Cited on pages 3, 23, 38, 49, 91 and 95.)

Wang, Xiaolong and Gupta, Abhinav. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2015. (Cited on page 13.)

Wang, Xiaolong; Jabri, Allan, and Efros, Alexei A. Learning correspondence from the cycle-consistency of time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019c. (Cited on page 13.)

Wang, Zhenzhi; Gao, Ziteng; Wang, Limin; Li, Zhifeng, and Wu, Gangshan. Boundary-aware cascade networks for temporal action segmentation. In *Proceedings of the European Conference on Computer Vision*, 2020c. (Cited on pages 26, 111 and 113.)

Wei, Donglai; Lim, Joseph J.; Zisserman, Andrew, and Freeman, William T. Learning and using the arrow of time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. (Cited on page 12.)

Weinberger, Kilian Q; Blitzer, John, and Saul, Lawrence. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, 2006. (Cited on page 74.)

Winkens, Jim; Bunel, Rudy; Roy, Abhijit Guha; Stanforth, Robert; Natarajan, Vivek; Ledsam, Joseph R.; MacWilliams, Patricia; Kohli, Pushmeet; Karthikesalingam, Alan; Kohl, Simon; Cemgil, A. Taylan; Eslami, S. M. Ali, and Ronneberger, Olaf. Contrastive training for improved out-of-distribution detection. *arXiv*, abs/2007.05566, 2020. (Cited on pages 18, 76, 77 and 78.)

Wiskott, Laurenz and Sejnowski, Terrence J. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 2002. (Cited on page 13.)

Wu, Yuxin and He, Kaiming. Group normalization. In *Proceedings of the European Conference on Computer Vision*, 2018. (Cited on page 22.)

Xian, Yongqin; Lampert, Christoph H.; Schiele, Bernt, and Akata, Zeynep. Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. (Cited on page 79.)

Xiao, Fanyi; Tighe, Joseph, and Modolo, Davide. MoDist: Motion distillation for self-supervised video representation learning. *arXiv*, abs/2106.09703, 2021a. (Cited on page 21.)

Xiao, Tete; Wang, Xiaolong; Efros, Alexei A, and Darrell, Trevor. What should not be contrastive in contrastive learning. In *International Conference on Learning Representations*, 2021b. (Cited on page 16.)

Xie, Saining; Sun, Chen; Huang, Jonathan; Tu, Zhuowen, and Murphy, Kevin. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision*, 2018. (Cited on page 23.)

Xu, Dejing; Xiao, Jun; Zhao, Zhou; Shao, Jian; Xie, Di, and Zhuang, Yueting. Self-supervised spatiotemporal learning via video clip order prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. (Cited on pages 12, 55, 56, 92, 96 and 97.)

Xu, Hu; Ghosh, Gargi; Huang, Po-Yao; Okhonko, Dmytro; Aghajanyan, Armen; Metze, Florian; Zettlemoyer, Luke, and Feichtenhofer, Christoph. VideoCLIP: Contrastive pre-training for zero-shot video-text understanding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021. (Cited on page 21.)

Yang, Ceyuan; Xu, Yinghao; Dai, Bo, and Zhou, Bolei. Video representation learning with visual tempo consistency. *arXiv*, abs/2006.15489, 2020. (Cited on page 19.)

Yao, Ting; Zhang, Yiheng; Qiu, Zhaofan; Pan, Yingwei, and Mei, Tao. SeCo: Exploring sequence supervision for unsupervised representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. (Cited on pages 19 and 20.)

Yao, Yuan; Liu, Chang; Luo, Dezhao; Zhou, Yu, and Ye, Qixiang. Video playback rate perception for self-supervised spatio-temporal representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. (Cited on pages 14, 92 and 97.)

Yi, Fangqiu; Wen, Hongyu, and Jiang, Tingting. ASFormer: Transformer for action segmentation. In *British Machine Vision Conference*, 2021. (Cited on pages 26, 44, 45, 106, 107, 111, 112, 113, 114, 115, 119 and 120.)

Yosinski, Jason; Clune, Jeff; Bengio, Yoshua, and Lipson, Hod. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, 2014. (Cited on pages 36 and 53.)

Yue-Hei Ng, Joe; Hausknecht, Matthew; Vijayanarasimhan, Sudheendra; Vinyals, Oriol; Monga, Rajat, and Toderici, George. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015. (Cited on page 22.)

Zbontar, Jure; Jing, Li; Misra, Ishan; LeCun, Yann, and Deny, Stéphane. Barlow twins: Self-supervised learning via redundancy reduction. In *Proceedings of the International Conference on Machine Learning*, 2021. (Cited on page 22.)

Zhang, Can; Yang, Tianyu; Weng, Junwu; Cao, Meng; Wang, Jue, and Zou, Yuexian. Unsupervised pre-training for temporal action localization tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. (Cited on page 18.)

Zhang, Richard; Isola, Phillip, and Efros, Alexei A. Colorful image colorization. In *Proceedings of the European Conference on Computer Vision*, 2016. (Cited on page 14.)

Zhao, Nanxuan; Wu, Zhirong; Lau, Rynson W. H., and Lin, Stephen. What makes instance discrimination good for transfer learning? In *International Conference on Learning Representations*, 2021. (Cited on pages 64 and 67.)

Zhu, Rui; Zhao, Bingchen; Liu, Jingen; Sun, Zhenglong, and Chen, Chang Wen. Improving contrastive learning by visualizing feature transformation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. (Cited on page 17.)

Zhu, Yi; Li, Xinyu; Liu, Chunhui; Zolfaghari, Mohammadreza; Xiong, Yuanjun; Wu, Chongruo; Zhang, Zhi; Tighe, Joseph; Manmatha, R., and Li, Mu. A comprehensive study of deep video action recognition. *arXiv*, abs/2012.06567, 2020. (Cited on page 22.)

Zhuang, Chengxu; Zhai, Alex Lin, and Yamins, Daniel. Local aggregation for unsupervised learning of visual embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. (Cited on page 20.)

Zhuang, Chengxu; She, Tianwei; Andonian, Alex; Mark, Max Sobol, and Yamins, Daniel. Unsupervised learning from video with deep neural embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. (Cited on pages 20, 56, 81, 82 and 92.)

Zolfaghari, Mohammadreza; Zhu, Yi; Gehler, Peter, and Brox, Thomas. CrossCLR: cross-modal contrastive learning for multi-modal video representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. (Cited on page 21.)