Self-supervised Video Representation Learning

Tengda Han

Lady Margaret Hall
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity 2022

Abstract

Videos are an appealing source of data to train computer vision models. There exist almost infinite supplies of videos online, but exhaustive manual annotation is infeasible. The goal of this thesis is to learn strong video representations efficiently via self-supervised learning: a method that learns from the data rather than human annotations.

The thesis is structured around three themes: (1) self-supervised learning for short-term videos, (2) efficient video representation learning, and (3) self-supervised learning for long-term videos.

For short-term videos lasting only a few seconds, we show that predicting the video in the future is a strong learning signal at a large scale. We further show that strong video representations can be learned by taking two complementary modalities, namely RGB and optical flow, and using them to teach each other.

For efficient video representation learning, we show that large-scale pre-trained vision-language models can be effectively adapted via a prompt tuning technique. We also show that dropping image patches can accelerate the finetuning of classification tasks and pre-training of video-language models.

For long-term videos that last longer than a few minutes, we show that temporal alignment networks can be trained from the weak visual-textual correspondence within instructional videos. The resulting networks can automatically clean up the natural videos for effective vision-language training. In addition, we show that movie description models can be trained by leveraging the pre-trained vision-language models.

Keywords – video understanding, deep learning, self-supervision, efficient learning

This thesis is submitted to the Department of Engineering Science, The University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Tengda Han, October 2022.

Acknowledgement

First and foremost to my supervisor Andrew Zisserman, thank you for your guidance, advice and trust along this journey. I feel extremely fortunate to have worked with you. You are my role model in so many aspects. Similarly to Weidi Xie, my wonderful co-supervisor and long-term collaborator, thank you for your intuition and in-depth guidance. The support from you two makes this experience much more enjoyable.

To all the VGG members: thank you for making VGG such a lovely group, of which I feel privileged to be a member. In particular to Andrew Brown, Shangzhe Wu and Chuhan Zhang for the great times we spent together. To Max Bain, Gül Varol and Arsha Nagrani for the great collaboration. To Yuki Asano for insightful discussions and wine tasting in Porto. Especially to Triantafyllos Afouras, who bought me so much beer in the pubs over the years, thank you for your great company. I also thank Robert McCraith, Hala Lamdouar, Liliane Momeni, Charig Yang and other VGG members for their support and camaraderie. Importantly, I thank Ashish Thandavan for answering my endless server questions and Jenny Hu for crucial group management efforts.

I also thank my friends outside of VGG: Yang Zhang, Zirui Wang and Yizhe Wu, for the laughter on the football courts and foosball tables. Berkeley friends Allan Jabri and Sheng-Yu Wang, for the wonderful memories on the river and with a guitar during my first year. Richard Hartley, for kindly introducing me to VGG and Oxford at the beginning. Jean-Baptiste Alayrac, Karen Simonyan and the entire Flamingo team at Deepmind, for their kindness and patience during my internship.

Thank Deepmind for generously funding my DPhil programme. Also, I thank my college Lady Margaret Hall for the pianos, library, lively gardens and the River Cherwell.

Finally, I thank my parents for their understanding and support, especially during the pandemic. Lastly to CZ, for lighting up my life.

Contents

Chapter 1

Introduction and Background

Videos are the media that locks both space and time with bits; they contain the prominent human sensory information like visual and audio; and they also carry rich information about human cognition like reasoning, knowledge, and decision-making, which reflect human intelligence. Understanding videos is therefore an important milestone in achieving artificial intelligence. Fortunately, videos are now massively created and shared on the internet, due to advanced network and storage technologies, and have become an infinite data source that can be used to train computer vision systems. However, the appealing enormous availability of videos rules out the possibilities of manual data annotation, and drives researchers to create methods to learn from the raw data itself; an approach that is termed 'self-supervised learning'.

In contrast to the supervised learning that learns from the 'sample - label' pairs (e.g. [Krizhevsky et al. 2012]), self-supervised learning (SSL) learns from a task without involving any manually-defined labels. This task is often called a 'proxy task' that only provides learning signals but is not the ultimate goal of learning. After learning from the proxy task, the model produces *representations* of the input samples. By that, it means the input samples are represented as vectors with finite dimensions. The quality of representation learning is typically evaluated by adapting the learned representations to downstream tasks of interest, like classification, segmentation and retrieval.

In this thesis, we address the methodologies for self-supervised video representation learning in a modern point of view. We ask the question, *'How can a strong video representation be learned from large-scale data, without human an-*

notations, and efficiently?' Particularly, we tackle three aspects of this question: self-supervised learning for short-term videos, efficient self-supervised training, and self-supervised learning for long-term videos.

We begin the thesis by introducing motivations and key ideas behind the work in Section 1.1, then we introduce the three themes of the thesis and their respective contributions in Section 1.2, finally in Section 1.3 we provide a full list of the works included in the thesis.

1.1 Motivation and Key Ideas

1.1.1 Motivation from Psychology

The Nature of Video. From daily experience, humans perceive continuous visual signals over time. Psychologists think visual continuity is critical to human cognitive development, e.g. infants acquire visual tracking in their first month [Piaget 1977]. Whereas videos display a series of still images, how do they look so natural? The persistence of human vision is a property that the visual perception of an object continues for a short time after the light rays proceeding from it have stopped to enter the eye [Nichol 1857]. Using this property, videos achieve a comfortable illusion of continuously moving scenes by refreshing the still images at a frequency above certain thresholds [Watson 1986], which mimics the visual sensory input from the real world.

Two-stream Hypothesis. For the human visual system, two-stream hypothesis [Goodale and Milner 1992] states that the human visual cortex contains two pathways – the ventral stream that recognises objects, and the dorsal stream that recognises motion.

Multiple Modalities. Apart from visual inputs, humans take multiple sensory inputs like audio, touch, smell, *etc.* Psychologists found that multiple sensory streams have a high redundancy [Smith et al. 1998], such that the entire perceptual system can function robustly even with the absence of one sensory stream. Also, Psychologists found a mechanism called *re-entry* [Edelman and Gally 2013], that the multi-sensory signals are exchanged continuously in a bi-directional way

between brain regions, thus the stimuli from one sensory stream can immediately trigger other sensory streams.

Predictive Coding. Possibly dated back to [von Helmholtz 1867], predictive coding is a theory in neuroscience about the brain function in which the brain is constantly generating predictions of sensory inputs to compare with the actual sensory inputs. Such a mechanism ensures a mental model of the environment is continuously adapting and improving. [Rao and Ballard 1999] framed the theory into a computational model of human vision that consists of (1) a generation model that produces a top-down prediction of the scene and (2) an analytical model that performs bottom-up sensory-driven error analysis.

1.1.2 Key Ideas

Inspired by the psychological motivations, as well as influenced by other practical factors like hardware and implementation, we introduce a few key research ideas in this section that govern the thesis and guide the design of methods.

1.1.3 Learn from the Abundance of Video Data

Video is one of the most abundant visual data available, e.g., more than 720,000 hours of videos are uploaded to the video platform YouTube every day[1], which takes a person 82 years to watch them. We are interested in developing learning methods that benefit from the enormous amount of video data, because these methods have a high potential in the future given the rapid advancements in computational power and data processing. As such, large-scale training on natural videos is the main focus of this thesis. Despite limited computational resources, we explore large-scale training at our extreme on the big video datasets like Kinetics [Kay et al. 2017; João Carreira et al. 2018; João Carreira et al. 2019] and HowTo100M [Miech et al. 2019]. For method design, We discover learning signals from the data themselves, such that our methods can be scaled-up at ease given extra computational resources.

[1] https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/

1.1.4 Learn from Multiple Modalities

Videos naturally contain multiple modalities like audio (the sound from objects and environment, or human voice) and text (the video titles, subtitles and audience's comments). Within visual sensory inputs, the appearance and motion (optical flow) can also be loosely regarded as two modalities. As per psychological studies, multiple modalities are typically redundant yet complementary to each other, that is, different modalities represent different aspects of the same observation. For practical self-supervised learning, the connection between modalities can be used as a powerful learning signal. In our works, we use motion modality to teach appearance modality and vice-versa. We also rely on the co-occurrence of visual and textual modalities to learn powerful representations.

1.1.5 Learn Efficiently

Efficient learning is important for realising self-supervised learning at a large scale for two reasons. First, it democratises SSL research and application for communities with limited resource access. Second, it reduces the energy consumption and the negative environmental impact caused by large-scale computation [Strubell et al. 2019].

Therefore, efficient learning is another key idea of this thesis. We mainly apply three strategies to pursue efficient learning: (1) learning from pre-trained models to save unnecessary computational costs. (2) reducing the redundant or noisy learning signals, like the patches in consecutive video frames and irrelevant video subtitles. (3) steering big models by a small number of parameters, like prompt-tuning.

1.2 Thesis Outline and Contributions

This section provides an outline of the following thesis chapters and summarises the key contributions. The thesis is grouped into three themes: self-supervised learning for short-term videos, efficient video representation learning, and self-supervised learning for long-term videos.

Part I: Self-supervised learning for short-term videos

By the word 'short-term', we refer to the videos with less than 10 seconds in duration. The 10 seconds is a rough boundary taken from the seminal Kinetics dataset [Kay et al. 2017], where video clips are trimmed to around 10-second long. Within the limited duration, short-term video contains simpler semantic information without much variance in time, typically only sufficient to cover one action.

In this theme, we show that strong video representations can be learned by predicting the future and from complementary modalities. Particularly in Chapter 2, we are motivated by the intuition that predicting in pixel space is heavy-weight and unnecessary for learning higher-level semantics, then we propose our method that predicts future video in the feature space. Trained with a contrastive loss on spatio-temporal features, such a predictive model learns a powerful video representation for action classification tasks. Also predicting future videos is a universal task that applicable to any video data and it is shown to benefit from large-dataset training. However, we notice the property of multiple hypotheses in video predictions, i.e., multiple reasonable futures could happen given the present observation. As a consequence, models trained to make a single prediction converges to predict a mean value in feature space for multiple possible futures, which limits the learning progress. To tackle the multi-hypothesis problem, in Chapter 3 we augment the future prediction module with an external memory bank that consists of learnable feature vectors, in order to structurally support the learning with multiple futures. In Chapter 4, Utilising the multi-modality property of videos, we propose to use the Nearest Neighbours of one modality as the pseudo labels to train the other modality, and vice versa. Particularly, we experiment with the RGB video stream and the optical flow stream, and we show that the final representation quality for both modalities is largely improved.

Part II: Efficient video representation learning

From the practical experience from Part I, we notice that self-supervised learning for videos benefits from large data and long training schedules. However, considering the high resource demands and energy consumption of SSL training, developing efficient learning methods becomes critical.

In this theme, we show two methods that can reduce the training cost of video representation learning. In Chapter 5, we adapt the light-weight prompt-tuning method to the industry-scale pre-trained visual language model to enable a variety of downstream video understanding tasks with small computational costs, yet maintain comparable performances. In Chapter 6, we propose to drop out the redundant input visual patches of Vision Transformer [Dosovitskiy et al. 2020] for the video-language pre-training, video classification finetuning and long-video classification tasks. This technique could speed up the training process by up to 4× and enable those resource-demanding training tasks with limited resources.

Part III: Self-supervised learning for long-term videos

The vast majority of video data online are the un-trimmed long videos that span a few minutes or hours. Distinct from the short-term videos in Part I, the longer temporal axis contains rich information such as the procedures of a complex activity or the storyline of a film.

In this theme, two methods for long-term videos are shown, one learns from the instructional videos and the other learns from featured movies. In Chapter 7, we utilise the weak textual-visual correspondence from natural instructional videos on YouTube and propose a method to automatically align demonstrators' subtitles with visual content in the video. As a result, we obtain a Temporal Alignment Network that can clean up the natural instructional videos by performing an automatic textual-visual alignment. In Chapter 8, we focus on the featured movies that could span a few hours. Benefiting from the pre-trained vision-language models, we develop a method to generate audio descriptions for movies densely along the time axis.

1.3 Publications

Chapter 2 to 8 each contains a research paper. Most of them have been peer-reviewed and accepted for publication at a conference, with the exception that Chapter 8 is an ongoing work. We make no modifications to the published papers except for formatting changes. For each paper, we also provide a statement of authorship in Appendix A. The papers included in the thesis are listed below.

Chapter 2: "Video Representation Learning by Dense Predictive Coding" **Tengda Han**, Weidi Xie, Andrew Zisserman. In the Workshop on Large Scale Holistic Video Understanding (Oral Presentation), *International Conference on Computer Vision*, 2019.

Chapter 3: "Memory-augmented Dense Predictive Coding for Video Representation Learning" **Tengda Han**, Weidi Xie, Andrew Zisserman. In *European Conference on Computer Vision* (Spotlight), 2020.

Chapter 4: "Self-supervised Co-Training for Video Representation Learning" **Tengda Han**, Weidi Xie, Andrew Zisserman. In *Neural Information Processing Systems*, 2020.

Chapter 5: "Prompting Visual-Language Models for Efficient Video Understanding" Chen Ju, **Tengda Han**, Kunhao Zheng, Ya Zhang, Weidi Xie. In *European Conference on Computer Vision*, 2022.

Chapter 6: "Turbo Training with Token Dropout" **Tengda Han**, Weidi Xie, Andrew Zisserman. In *British Machine Vision Conference*, 2022.

Chapter 7: "Temporal Alignment Networks for Long-term Video" **Tengda Han**, Weidi Xie, Andrew Zisserman. In *Conference on Computer Vision and Pattern Recognition* (Oral Presentation), 2022.

Chapter 8: "AutoAD: Movie Description in Context" **Tengda Han**, Max Bain, Gül Varol, Arsha Nagrani, Weidi Xie, Andrew Zisserman. Technical report, 2022.

Publications not included

†**"Flamingo: a Visual Language Model for Few-Shot Learning"** Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, **Tengda Han**, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew

Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, Karen Simonyan. In *Neural Information Processing Systems*, 2022.

† This publication mainly focuses on image-language tasks and few-shot learning settings, which is loosely related to the thesis topic about video representation learning, therefore excluded.

Part I

Self-supervised Learning for Short-term Videos

Chapter 2

Video Representation Learning by Dense Predictive Coding

The paper has been accepted for publication as an oral presentation at the Workshop on Large-scale Holistic Video Understanding, International Conference on Computer Vision (ICCV), 2019.

Video Representation Learning by Dense Predictive Coding

Tengda Han Weidi Xie Andrew Zisserman

Visual Geometry Group, Department of Engineering Science, University of Oxford

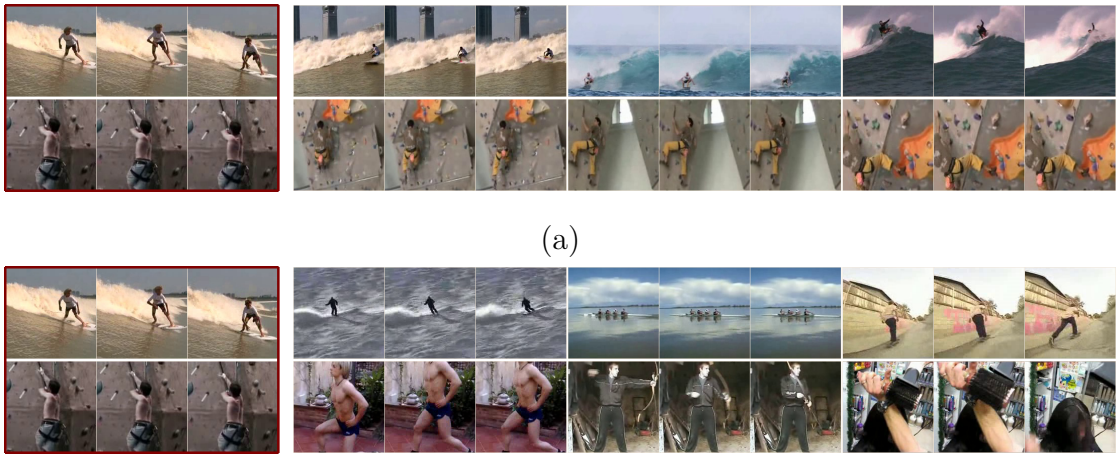
{htd, weidi, az}@robots.ox.ac.uk

https://www.robots.ox.ac.uk/~vgg/research/DPC/dpc.html

Abstract

The objective of this paper is self-supervised learning of spatio-temporal embeddings from video, suitable for human action recognition.

We make three contributions: First, we introduce the Dense Predictive Coding (DPC) framework for self-supervised representation learning on videos. This learns a dense encoding of spatio-temporal blocks by recurrently predicting future representations; Second, we propose a curriculum training scheme to predict further into the future with progressively less temporal context. This encourages the model to only encode slowly varying spatial-temporal signals, therefore leading to semantic representations; Third, we evaluate the approach by first training the DPC model on the Kinetics-400 dataset with self-supervised learning, and then fine-tuning the representation on a downstream task, i.e. ction recognition. With single stream (RGB only), DPC pretrained representations achieve state-of-the-art self-supervised performance on both UCF101 (75.7% top1 acc) and HMDB51 (35.7% top1 acc), outperforming all previous learning methods by a significant margin, and approaching the performance of a baseline pre-trained on ImageNet. The code is available at https://github.com/TengdaHan/DPC.



(a)

(b)

Figure 2.1: Nearest Neighbour (NN) video clip retrieval on UCF101. Each row contains four video clips, a query clip and the top three retrievals using clip embeddings. To get the embedding, each video is passed to a 3D-ResNet18, average pooled to a single vector, and cosine similarity is used for retrieval. (a) Embeddings obtained by Dense Predictive Coding (DPC); (b) Embeddings obtained by using the inflated ImageNet pretrained weights. The DPC captures the semantics of the human action, rather than the scene appearance or layout as captured by the ImageNet trained embeddings. In the DPC retrievals the actual appearances of frames can vary dramatically, e.g. in the change in camera viewpoint for the climbing case.

2.1 Introduction

Videos are very appealing as a data source for self-supervision: there is almost an infinite supply available (from Youtube etc.); image level proxy losses can be used at the frame level; and, there are plenty of additional proxy losses that can be employed from the temporal information. One of the most natural, and consequently one of the first video proxy losses, is to predict future frames in the videos based on frames in the past. This has ample scope for exploration by varying the extent of the past knowledge (the temporal aggregation window used for the prediction) and also the temporal distance into the future for the predicted frames. However, future frame prediction does have a serious disadvantage – that the future is not deterministic – so methods may have to consider multiple hypotheses with multiple instance losses, or other distributions and losses over their predictions.

Previous approaches to future frame prediction in video [Lotter et al. 2017; Mathieu et al. 2016; Srivastava et al. 2015; Vondrick et al. 2016b; Vondrick et al. 2016a] can roughly be divided into two types: those that predict a reconstruction

of the actual frames [Lotter et al. 2017; Mathieu et al. 2016; Srivastava et al. 2015; Vondrick et al. 2016b]; and those that only predict the latent representation (the embedding) of the frames [Vondrick et al. 2016a]. If our goal of self-supervision is only to learn a representation that allows generalization for downstream discriminative tasks, e.g. action recognition in video, then it may not be necessary to waste model capacity on resolving the stochasticity of frame appearance in detail, e.g. appearance changes due to shadows, illumination changes, camera motion, etc. Approaches that only predict the frame embedding, such as Vondrick *et al.* [Vondrick et al. 2016a], avoid this potentially unnecessary task of detailed reconstruction, and use a mixture model to resolve the uncertainty in future prediction. Although not applied to videos (but rather to speech signals and images), the Contrastive Predictive Coding (CPC) model of Oord *et al.* [van den Oord et al. 2018] also learns embeddings, in their case by using a multi-way classification over temporal audio frames (or image patches), rather than the regression loss of [Vondrick et al. 2016a].

In this paper we propose a new idea for learning spatio-temporal video embeddings, that we term "Dense Predictive Coding" (DPC). The model is designed to predict the future representations based on the recent past [Wiskott and Sejnowski 2002]. It is inspired by the CPC [van den Oord et al. 2018] framework, and more generally by previous research on learning word embeddings [Mikolov et al. 2013a; Mikolov et al. 2013b; Mnih and Kavukcuoglu 2013]. DPC is also trained by using a variant of noise contrastive estimation [Gutmann and Hyvärinen 2010], therefore, in practice, the model has never been optimized to predict the exact future, it is only asked to solve a multiple choice question, i.e. ick the correct future states from lots of distractors. In order to succeed in this task, the model only needs to learn the shared semantics of the multiple possible future states, and this common/shared representation is the kind of invariance required in many of the vision tasks, e.g. action recognition in videos. In other words, the optimization objective will actually benefit from the fact that the future is not deterministic, and map the representation of all possible future states to a space that their embeddings are close. Concurrent work [Anand et al. 2019] applies similar method on reinforcement learning.

The contributions of this paper are three-fold: First, we introduce Dense Predictive Coding (DPC) framework for self-supervised representation learning on

videos, we task the model to predict the future embedding of the spatio-temporal blocks recurrently (as used in N-gram prediction). The model is trained to pick the "correct" future states from a pool of distractors, therefore treated as a multi-way classification problem. Second, we propose a curriculum training scheme that enables the model to gradually predict further in the future (up to 2 seconds) with progressively less temporal context, leading more challenging training samples, and preventing the model from using shortcuts such as optical flow; Third, we evaluate the approach by first training the DPC model on the Kinetics-400 [Kay et al. 2017] dataset using self-supervised learning, and then fine-tuning on action recognition benchmarks. Our DPC model achieves state-of-the-art self-supervised performance on both UCF101 (75.7% top1 acc) and HMDB51 (35.7% top1 acc), outperforming all previous single-stream (RGB only) self-supervised learning methods by a significant margin.

2.2 Related Work

Self-supervised learning from images. In recent years, methods for self-supervised learning on images have achieved an impressive performance in learning high-level image representations. Inspired by the variants of Word2vec [Bengio et al. 2003; Mikolov et al. 2013a; Mikolov et al. 2013b] that rely on predicting words from their context, Doersch *et al.* [Doersch et al. 2015] proposed the pretext task of predicting the relative location of image patches. This work spawned a line of work in context-based self-supervised visual representation learning methods, e.g. in [Noroozi and Favaro 2016]. In contrast to the context-based idea, another set of pretext tasks include carefully designed image-level classification, such as rotation [Gidaris et al. 2018] or pseudo-labels from clustering [Caron et al. 2018]. Another class of pre-text tasks is for dense predictions, e.g. image inpainting [Pathak et al. 2016], image colorization [Richard Zhang et al. 2016], and motion segmentation prediction [Pathak et al. 2017]. Other methods instead enforce structural constraints on the representation space [Noroozi et al. 2017].

Self-supervised learning from videos. Other than the predictive tasks reviewed in the introduction, another class of proxy tasks is based on temporal sequence ordering of the frames [Misra et al. 2016; Fernando et al. 2017; Wei et

al. 2018]. [Isola et al. 2015; Jayaraman and Grauman 2016; X. Wang and Abhinav Gupta 2015] use the temporal coherence as a proxy loss. Other approaches use egomotion [P. Agrawal et al. 2015; Jayaraman and Grauman 2015] to enforce equivariance in feature space [Jayaraman and Grauman 2015]. In contrast, [Jing and Yingli Tian 2018] predicts the transformation applied to a spatio-temporal block. In [Kim et al. 2019], the authors propose to use a 3D puzzle as the proxy loss. Recently [Vondrick et al. 2018; Lai and W. Xie 2019; X. Wang et al. 2019], leveraged the natural temporal coherency of color in videos, to train a network for tracking and correspondence related tasks.

Action recognition with two-stream architectures. Recently, the two-stream architecture [Simonyan and Zisserman 2014] has been a foundation for many competitive methods. The authors show that optical flow is a powerful representation that improves action recognition dramatically. Other modalities like audio signal can also benefits visual representation learning [Korbar et al. 2018]. While in this paper, we deliberately avoid using any information from optical flow or audio, and aim to probe the upperbound of self-supervised learning with *only* RGB streams. We leave it as a future work to explore how much boost optical flow branch and audio branch can bring to our self-supervised learning architecture.

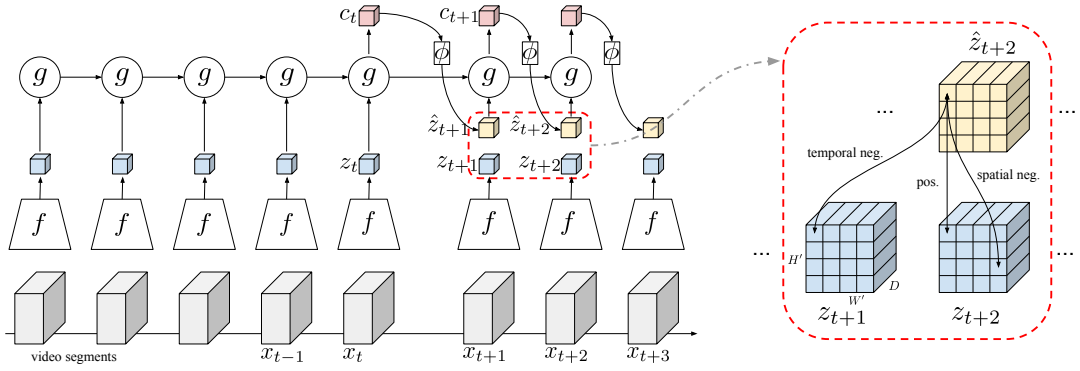


Figure 2.2: A diagram of **Dense Predictive Coding** method. The left part is the pipeline of the DPC, which is explained in Sec. 2.3.1. The right part (in the dashed rectangle) is an illustration of the Pred-GT pair construction for contrastive loss, which is explained in Sec. 2.3.2.

2.3 Dense Predictive Coding (DPC)

In this section, we describe the learning framework, details of the architecture, and the curriculum training that gradually learns to predict further into the future with progressively less temporal context.

2.3.1 Learning Framework

The goal of DPC is to predict a slowly varying semantic representation based on the recent past, e.g. we construct a prediction task that observes about 2.5 seconds of the video and predict the embedding for the future 1.5 seconds, as illustrated in Figure 2.2. A video clip is partitioned into multiple non-overlapping blocks $x_1, x_2, \ldots, x_n$, with each block containing an equal number of frames. First, a non-linear encoder function $f(.)$ maps each input video block x_t to its latent representation z_t, then an aggregation function $g(.)$ temporally aggregates t consecutive latent representations into a context representation c_t:

$$z_t = f(x_t) \tag{2.1}$$

$$c_t = g(z_1, z_2, ..., z_t) \tag{2.2}$$

where x_t has dimension $\mathbb{R}^{T\times H\times W\times C}$, and z_t is a feature map with dimension $\mathbb{R}^{1\times H'\times W'\times D}$, organized as time $\times$ height $\times$ width $\times$ channels. [1]

The intuition behind the predictive task is that if one can infer future semantics from c_t, then the context representation c_t and the latent representations $z_1, z_2, ..., z_t$ must have encoded strong semantics of the input video clip. Thus, we introduce a predictive function $\phi(.)$ to predict the future. In detail, $\phi(.)$ takes the context representation as the input and predicts the future clip representation:

$$\hat{z}_{t+1} = \phi(c_t) = \phi\Big(g(z_1, z_2, \ldots, z_t)\Big) \tag{2.3}$$

$$\hat{z}_{t+2} = \phi(c_{t+1}) = \phi\Big(g(z_1, z_2, \ldots, z_t, \hat{z}_{t+1})\Big) \tag{2.4}$$

where c_t denotes the context representation from time step 1 to t, and $\hat{z}_{t+1}$ denotes the predicted latent representation of the time step $t+1$. In the spirit of Seq2seq [Sutskever et al. 2014], representations are predicted in a sequential manner. We predict q steps in the future, at each time step t, the model consumes the previously generated embedding ($\hat{z}_{t-1}$) as input when generating the next ($\hat{z}_t$), further enforcing the prediction to be conditioned on all previous observations and predictions, and therefore encourages an N-gram like video representation.

[1] In our initial experiments, $x_t \in \mathbb{R}^{5\times 128\times 128\times 3}$, $z_t \in \mathbb{R}^{1\times 4\times 4\times 256}$

2.3.2 Contrastive Loss

Noise Contrastive Estimation (NCE) [Gutmann and Hyvärinen 2010] constructs a binary classification task: a classifier is fed with real samples and noise samples, and the objective is to distinguish them. A variant of NCE [Mnih and Kavukcuoglu 2013; van den Oord et al. 2018] classifies one real sample among many noise samples. Similar to [Mnih and Kavukcuoglu 2013; van den Oord et al. 2018], we use a loss based on NCE for the predictive task. NCE over feature embeddings encourages the predicted representation $\hat{z}$ to be close to the ground truth representation z, but not so strictly that it has to resolve the low-level stochasticity.

In the forward pass, the ground truth representation z and the predicted representation $\hat{z}$ are computed. The representation for the i-th time step is denoted as z_i and $\hat{z}_i$, which have the same dimensions. Note that, instead of pooling into a feature vector, both z_i and $\hat{z}_i$ are kept as feature maps ($z_i, \hat{z}_i \in \mathbb{R}^{H' \times W' \times D}$), which maintains the spatial layout representation. We denote the feature vector in each spatial location of the feature map as $z_{i,k} \in \mathbb{R}^D$ and $\hat{z}_{i,k} \in \mathbb{R}^D$ where i denotes the temporal index and k is the spatial index $k \in \{(1,1),(1,2),\ldots,(H,W)\}$. The similarity of the predicted and ground-truth pair (Pred-GT pair) is computed by the dot product $\hat{z}_{i,k}^\top z_{j,m}$. The objective is to optimize:

$$\mathcal{L} = -\sum_{i,k}\left[\log \frac{\exp(\hat{z}_{i,k}^\top \cdot z_{i,k})}{\sum_{j,m}\exp(\hat{z}_{i,k}^\top \cdot z_{j,m})}\right] \tag{2.5}$$

In essense, this is simply a cross-entropy loss (negative log-likelihood) that distinguishes the positive Pred-GT pair out of all other negative pairs. For a predicted feature vector $\hat{z}_{i,k}$, the only positive pair is $(\hat{z}_{i,k}, z_{i,k})$, i.e. the predicted and ground-truth features at the same time step and same spatial location. All the other pairs $(\hat{z}_{i,k}, z_{j,m})$ where $(i,k) \neq (j,m)$, are negative pairs. The loss encourages the positive pair to have a higher similarity than any negative pairs. If the network is trained in a mini-batch consisting of B video clips and each of the B clips is from distinct video, more negative pairs can be obtained.

To discriminate the different types of negative pairs, given a Pred-GT pair $(\hat{z}_{i,k}, z_{j,m})$, we define the terminology as follows:

Easy negatives: is the Pred-GT pair that is formed from two distinct videos. These pairs are naturally easy because they usually have distinct color distributions and thus predicted feature and ground-truth feature have low similarity.

Spatial negatives: is the Pred-GT pair that is formed from the same video but at a different spatial position in the feature map, i.e. $(k \neq m)$, while i, j can be any index.

Temporal negatives (hard negatives): is the Pred-GT pair that comes from the same video and same spatial position, but from different time steps, i.e. $(k = m, i \neq j)$. They are the hardest pair to classify because their score will be very close to the positive pairs.

Overall, we use a similar idea to the Multi-batch training [Tadmor et al. 2016]. If the mini-batch has batch size B, the feature map has spatial dimension $H' \times W'$ and the task is to classify one of q time steps, the number of each classes follows:

$$\begin{aligned} &\text{Pos} : \text{N}_{temporal} : \text{N}_{spatial} : \text{N}_{easy} \\ =&1 : (q-1) : (H'W'-1)q : (B-1)H'W'q \end{aligned}$$

Curriculum learning strategy. A curriculum learning strategy is designed by progressively increasing the number of prediction steps of the model (Sec. 2.4.1). For instance, the training process can start by predicting only 2 steps (about 1 second), i.e. nly computing $\hat{z}_{t+1}$ and $\hat{z}_{t+2}$, and the Pred-GT pairs are constructed between $\{z_{t+1}, z_{t+2}\}$ and $\{\hat{z}_{t+1}, \hat{z}_{t+2}\}$. After the network has learnt this simple task, it can be trained to predict 3 steps (about 1.5 seconds), e.g. computing $\hat{z}_{t+1}$, $\hat{z}_{t+2}$ and $\hat{z}_{t+3}$ and construct Pred-GT pairs accordingly. Importantly, curriculum learning introduces more hard negatives throughout the training process, and forces the model to gradually learn to predict further in the future with progressively less temporal context. Meanwhile, the model is gradually trained to grasp the uncertain nature in its prediction.

2.3.3 Avoiding Shortcuts and Learning Semantics

Empirical experience in self-supervised learning indicates that if the proxy task is well-designed and requires semantic understanding, a more difficult learning task usually leads to a better-quality representation [Lee et al. 2017]. However,

ConvNets are notoriously known for learning shortcuts for tackling tasks [Doersch et al. 2015; Noroozi and Favaro 2016; Wei et al. 2018]. In our training, we employ a number of mechanisms to avoid potential shortcuts, as detailed next.

Disrupting optical flow. A trivial solution of our predictive task is that $f(.)$, $g(.)$ and $\phi(.)$ together learn to capture low-level optical flow information and perform feature extrapolation as the prediction. To force the model to learn high-level semantics, a critical operation is frame-wise augmentation, i.e. andom augmentation for each individual frame in the video blocks, such as frame-wise color jittering including random brightness, contrast, saturation, hue and random greyscale during training. Furthermore, the curriculum of predicting further into the future, i.e. redicting the semantics for the next a few seconds, also ensures that optical flow alone will not be able to solve this prediction task.

Temporal receptive field. The temporal receptive field (RF) of $f(.)$ is limited by cutting the input video clip into non-overlapping blocks before feeding it into $f(.)$. Thus, the effective temporal RF of each feature map z_i is strictly restricted to be within each video block. This avoids the network being able to discriminate positive and hard-negative by recognizing relative temporal position.

Spatial receptive field. Due to the depth of CNN, each feature vector $\hat{z}_{i,k}$ in the final predicted feature map $\hat{z}_i$ has a large spatial RF that (almost) covers the entire input spatial dimension. This creates a shortcut to discriminate positive and spatial negative by using padding patterns. One can limit the spatial RF by cutting input frames into patches [van den Oord et al. 2018; Kim et al. 2019]. However this brings some drawbacks: First, the self-supervised pre-trained network will have limited receptive field (RF), so the representation may not generalize well for downstream tasks where a large RF is required. Second, limiting spatial RF in videos makes the context feature too weak. The context feature has a spatio-temporal RF that covers a thin cube in the video flow. Neglecting context is also not ideal for understanding video semantics and brings ambiguity to the predictive task. Considering this trade-off, our method does not restrict the spatial RF.

Batch normalization. Common practice uses Batch Normalization [Ioffe and Szegedy 2015] (BN) in deep CNN architecture. The BN layer may provide shortcuts that the network acknowledges the statistical distribution of the mini-batch, which benefits the classification. In [van den Oord et al. 2018], the authors demon-

strate BN results in network cheating, and the ResNet trained with BN does not generalize to the downstream image classification task. In our method, we find the effect of BN shortcut is very limited. The self-supervised training gives similar accuracy using either BN or Instance Normalization [Ulyanov et al. 2016] (IN). For downstream tasks like classification, a network with BN gives 5%-10% accuracy gain comparing with a network with IN. It is hard to train a deep CNN without normalization for either self-supervised training or supervised training. Overall, we use BN in our encoder function $f(.)$.

2.3.4 Network Architecture

We choose to use a 3D-ResNet similar to [Hara et al. 2018] as the encoder $f(.)$. Following the convention of [Feichtenhofer et al. 2019a] there are four residual blocks in ResNet architecture, namely res_2, res_3, res_4 and res_5, and only expand the convolutional kernels in res_4 and res_5 to be 3D ones. For experiment analysis, we used 3D-ResNet18, denoted as R-18 below.

To train a strong encoder $f(.)$, a weak aggregation function $g(.)$ is preferable. Specifically, a one-layer Convolutional Gated Recurrent Unit (ConvGRU) with kernel size $(1, 1)$ is used, which shares the weights amongst all spatial positions in the feature map. This design allows the aggregation function to propagate features in the temporal axis. A dropout [Srivastava et al. 2014] with $p = 0.1$ is used when computing hidden state in each time step. A shallow two-layer perceptron is used as the predictive function $\phi(.)$.

2.3.5 Self-Supervised Training

For data pre-processing, we use 30 fps videos with a uniform temporal downsampling by factor 3, i.e. ake one frame from every 3 frames. These consecutive frames are grouped into 8 video blocks where each block consists of 5 frames. Frames are sampled in a consecutive way with consistent temporal stride to preserve the temporal regularity, because random temporal stride introduces uncertainties to the predictive task especially when the network needs to distinguish the difference among different time steps. Specifically, each video block spans over 0.5s and the entire 8 segments span over 4s in the raw video. The predictive task is initially designed to observe the first 5 blocks and predict the remaining 3 blocks (denoted

as '5pred3' afterwards), which is observing 2.5 seconds to predict the following 1.5 seconds. We also experiment with different predictive configuration like 4pred4 in Sec. 2.4.1.

For data augmentation, we apply random crop, random horizontal flip, random grey, and color jittering. Note that the random crop and random horizontal flip are applied for the entire clip in a consistent way. Random grey and color jittering are applied in a frame-wise manner to prevent the network from learning low-level flow information as mentioned above (in Sec. 2.3.3), e.g. each video block may contain both colored and grey-scale image with different contrast. All models are trained end-to-end using Adam [Kingma and Ba 2014b] optimizer with an initial learning rate 10^{-3} and weight decay 10^{-5}. Learning rate is decayed to 10^{-4} when validation loss plateaus. A batchsize of 64 samples per GPU is used, and our experiments use 4 GPUs.

2.4 Experiments and Analysis

In the following sections we present controlled experiments, and aim to investigate four aspects: *First*, an ablation study on the DPC model to show the function of different design choices, e.g. sequential prediction, dense prediction. *Second*, the benefits of training on a larger, and more diverse dataset. *Third*, the correlation between performance on self-supervised learning and performance on the downstream supervised learning task. *Fourth*, the variation in the learnt representations when predicting further into the future.

Datasets. The DPC is a general self-supervised learning framework for any video types, but we focus here on human action videos e.g. UCF101 [Soomro et al. 2012], HMDB51 [Hildegard Kuehne et al. 2011] and Kinetics-400 [Kay et al. 2017] datasets. UCF101 contains 13K videos spanning over 101 human action classes. HMDB51 contains 7K videos from 51 human action classes. Kinetics-400 (K400) is a big video dataset containing 306K video clips for 400 human action classes.

Evaluation methodology. The self-supervised model is trained either on UCF101 or K400. The representation is evaluated by its performance on a downstream task, i.e. ction classification on UCF101 and HMDB51. For all the experiments below: we report top1 accuracy for self-supervised learning in the middle column of all tables; and report the top1 accuracy for supervised learning for action classification

on UCF101 in the rightmost column. In self-supervised learning, the top1 accuracy refers to how often the multi-way classifier picks the right Pred-GT pair, i.e. his is not related with any action classes. While for supervised learning, the top1 accuracy indicates the action classification accuracy on UCF101. Note, we report the first training/testing splits of UCF101 and HMDB51 in all the experiments, apart from the comparison with the state of the art in Table 8.5 where we report the average accuracy over three splits.

Action classifier. During supervised learning, 5 video blocks are passed as input (the same as for self-supervised training, i.e. ach block is of $\mathbb{R}^{5\times128\times128\times3}$), and encoded as a sequence of feature maps with the encoding function $f(.)$ (a 3D-ResNet). As with the self-supervised architecture, the aggregation function $g(.)$ (a ConvGRU) aggregates the feature maps over time and produces a context feature. The context feature is further passed through a spatial pooling layer followed by a fully-connected layer and a multi-way softmax for action classification. The classifier is trained using the Adam [Kingma and Ba 2014b] optimizer with an initial learning rate 10^{-3} and weight decay 10^{-3}. Learning rate is decayed twice to 10^{-4} and 10^{-5} Note that the *entire* network is trained end-to-end. The details of the architecture are given in the appendix.

During inference, video clips from the validation set are densely sampled from an input video and cut into blocks ($\mathbb{R}^{5\times128\times128\times3}$) with half-length overlapping. Augmentations are removed and only center crop is used. The softmax probabilities are averaged to give the final classification result.

2.4.1 Performance Analysis

Ablation Study on Architecture

In this section, we present an ablation study by gradually removing components from the DPC model (see Table 2.1). For efficiency, all the self-supervised learning experiments refer to the 5pred3 setting, i.e. video blocks (2.5 second) are used as input to predict the future 3 steps (1.5 second).

Compared with the baseline model trained with random initialization and fully supervised learning, our DPC model pre-trained with self-supervised learning has a significant boost (top1 acc: 46.5% vs. 60.6%). When removing the sequential prediction, i.e. ll 3 future steps are predicted in parallel with three different fully-

Network	Self-Sup. (UCF)			Sup. (UCF)
	setting	method	top1 acc	top1 acc
R-18	-	- (rand. init.)	-	46.5
R-18	5pred3	DPC	**53.6**	**60.6**
R-18	5pred3	remove Seq.	51.3	56.9
R-18	5pred3	remove Map	36.5	44.9

Table 2.1: Ablation study of DPC. *remove Seq* means removing the sequential prediction mechanism in DPC, and replacing by parallel prediction. *remove Map* means removing the dense feature map design in DPC, and use a feature vector instead. Self-supervised tasks are trained on UCF101 using 5pred3 setting. Representation learned from each self-supervised task is evaluated by training a supervised action classifier on UCF101.

connected layers, the accuracy for both self-supervised learning and supervised learning start to drop. Lastly, we further replace the dense feature map by the average-pooled feature vector, i.e. t becomes a CPC-like model, we are not able to train this model either on self-supervised learning task or supervised learning. This demonstrates that *dense* predictive coding is essential to our success, and sequential prediction also helps to boost the model performance.

Benefits of Large Datasets

In this section, we investigate the benefits of pre-training on a large-scale dataset (UCF101 vs. K400), we keep the 5pred3 setting and evaluate the effectiveness for downstream task on UCF101. Results are shown in Table 2.2.

Network	Self-Sup.			Sup. (UCF)
	setting	dataset	top1 acc	top1 acc
R-18	5pred3	UCF101	53.6	60.6
R-18	5pred3	K400	**61.1**	**65.9**

Table 2.2: Results of DPC on UCF101 and K400 respectively. Both experiments use 5pred3 setting. Representations are evaluated by training a supervised action classifier on UCF101 (right column).

Training the model on K400 increases the self-supervised accuracy to 61.1%, and supervised accuracy from 60.6% to 65.9%, suggesting the model has captured more regularities than a smaller dataset like UCF101. It is clear that DPC will benefit from large-scale video dataset (infinite supply available), which naturally provides more diverse negative Pred-GT pairs.

Self-Supervised vs. Classification Accuracy

In this section, we investigate the correlation between the accuracy of self-supervised learning and downstream supervised learning. While training DPC (5pred3 task on K400), we evaluate the representation at different training stages (number of epochs) on the downstream task (on UCF101). The results are shown in Figure 2.3.

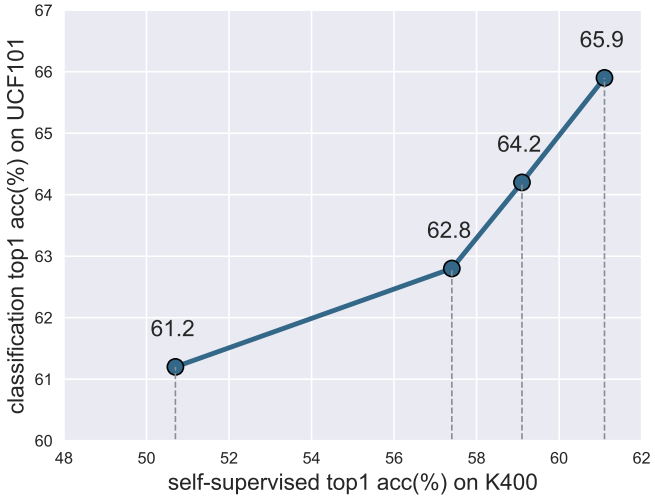


Figure 2.3: Relation between self-supervised accuracy and classification accuracy. Self-supervised model (DPC) is trained on K400 and the weights at epoch $\{13, 48, 81, 109\}$ are saved, which achieve $\{50.7\%, 57.4\%, 59.1\%, 61.1\%\}$ self-supervised accuracy respectively. The checkpoints are evaluated by finetuning on UCF101.

It can be seen that a higher accuracy in self-supervised task always leads to a higher accuracy in downstream classification. The result indicates that DPC has actually learnt visual representations that are not only specific to self-supervised task, but are also generic enough to be beneficial for the downstream task.

Benefits of Predicting Further into the Future

Due to the increase of uncertainty, predicting further into the future in video sequences gets more difficult, therefore more abstract (semantic) understanding is required. We hypothesize that if we can train the model to predict further, the learnt representation should be even better. In this section, we employ curriculum learning to gradually train the model to predict further with progressively less temporal context, i.e. rom 5pred3 to 4pred4 (4 video blocks as input and predict the future 4 steps).

The result shows that the 4pred4 setting gives a substantially lower accuracy on the self-supervised learning than 5pred3. This is actually not surprising, as 4pred4

Network	Self-Sup. (K400)			Sup. (UCF)
	setting	curr.	top1 acc	top1 acc
R-18	5pred3	✗	61.1	65.9
R-18	4pred4	✗	48.3	64.9
R-18	5pred3+4pred4	✓	**50.8**	**68.2**

Table 2.3: Results of DPC with different prediction steps. All models are trained on K400 with *same* number of 320k iterations. Note that for 5pred3 and 4pred4, the model is trained from scratch. '5pred3+4pred4' denotes that curriculum learning strategy, i.e. nitialized with the pre-trained weights from 5pred3 task. The representation is evaluated by training an action classifier on UCF101 (right column).

naturally introduces 33% more hard negative pairs than predicting future 3 steps, making the self-supervised learning more difficult (explained in Section 2.3.2).

Interestingly, despite a lower accuracy on self-supervised learning task, when comparing with 5pred3, curriculum learning on 4pred4 provides 2.3% performance boost on the downstream supervised task (top1 acc: 68.2% vs. 65.9%). The experiment also shows that curriculum learning is effective as it achieves higher performance than training 4pred4 task from scratch (top1 acc: 68.2% vs. 64.9%). Similar effect is also observed in [Korbar et al. 2018].

Summary

Through the experiments above, we have demonstrated the keys to the success of DPC. *First*, it is critical to do dense predictive coding, i.e. redicting both temporal and spatial representation in the future blocks, and sequential prediction enables a further boost in the quality of the learnt representation. *Second*, a large-scale dataset helps to improve the self-supervised learning, as it naturally contains more world patterns and provides more diverse negative sample pairs. *Third*, the representation learnt from DPC is generic, as a higher accuracy in the self-supervised task also yield a higher accuracy in the downstream classification task. *Fourth*, predicting further into the future is also beneficial, as the model is forced to encode the high-level semantic representations, and ignore the low-level information.

2.5 Comparison with State-of-the-art Methods

The results are given in Table 8.5, four phenomena can be observed: *First*, when self-supervised training with only UCF101, our DPC (60.6%) outperforms all previous methods under similar settings. Note that OPN [Lee et al. 2017] performs

Self-Supervised Method (RGB stream only)			Supervised Acc. (top1)	
Method	Architecture (#param)	Dataset	UCF101	HMDB51
Random Initialization	3D-ResNet18 (14.2M)	-	46.5	17.1
ImageNet Pretrained	VGG-M-2048 (25.4M)	-	73.0	40.5
Shuffle & Learn (227 × 227)	CaffeNet (58.3M)	UCF101/HMDB51	50.2	18.1
OPN (80 × 80)	VGG-M-2048 (8.6M)	UCF101/HMDB51	59.8	23.8
OPN (120 × 120)	VGG-M-2048 (11.2M)	UCF101/HMDB51	55.4	-
OPN (224 × 224)	VGG-M-2048 (25.4M)	UCF101/HMDB51	51.9	-
Ours (128 × 128)	3D-ResNet18 (14.2M)	UCF101	**60.6**	-
3D-RotNet (112 × 112)	3D-ResNet18-full (33.6M)	Kinetics-400	62.9	33.7
3D-ST-Puzzle (224 × 224)	3D-ResNet18-full (33.6M)	Kinetics-400	63.9 (65.8⋆)	33.7⋆
Ours (128 × 128)	3D-ResNet18 (14.2M)	Kinetics-400	**68.2**	**34.5**
Ours (224 × 224)	3D-ResNet34 (32.6M)	Kinetics-400	**75.7**	**35.7**

Table 2.4: Comparison with other self-supervised methods, results are reported as an average over three training-testing splits. Full reference list: ImageNet Pretrained [Simonyan and Zisserman 2014], Shuffle & Learn [Misra et al. 2016], OPN [Lee et al. 2017], 3D-RotNet [Jing and Yingli Tian 2018], 3D-ST-Puzzle [Kim et al. 2019]. Note that, previous works [Jing and Yingli Tian 2018; Kim et al. 2019] use full-scale 3D-ResNet18, i.e. ll convolutions are 3D, and the input sizes for different models have been shown. ⋆indicates the results from the multi-task self-supervised learning, i.e. otation + 3D Puzzle.

worse when input resolution increases, which indicates a simple self-supervised task like order prediction may not capture the rich semantics from videos. *Second*, when using Kinetics-400 for self-supervised pre-training, our DPC (68.2%) outperforms all the previous methods by a large margin. Note that, in the work [Jing and Yingli Tian 2018; Kim et al. 2019], the authors use a full-scale 3D-ResNet18 architecture (33.6M parameters), i.e. ll convolutions are 3D, however our modified 3D-ResNet18 has fewer parameters (only the last 2 blocks are 3D convolutions). The authors of [Kim et al. 2019] obtain 65.8% accuracy by combing the rotation classification [Jing and Yingli Tian 2018] with their Space-Time Cubic Puzzles method, essentially multi-task learning. When only considering their Space-Time Cubic Puzzles method, they obtain 63.9% top1 accuracy. On HMDB51, our method also outperforms the previous state of the art result by 0.8% (34.5% vs. 33.7%). *Third*, when applying on larger input resolution (224 × 224) and using model with more capacity (3D-ResNet34), our DPC clearly dominate all self-supervised learning methods (75.7% on UCF101 and 35.7% on HMDB51), further demonstrating that DPC is able to take advantage from networks with more capacity and today's large-scale datasets. *Fourth*, ImageNet pretrained weights have been a golden baseline for action recognition [Simonyan and Zisserman 2014], our self-supervised DPC is the first model that surpasses the performance of models (VGG-M) pre-trained with ImageNet (75.7% vs. 73.0% on UCF101).

2.5.1 Visualization

(a)

(b)

Figure 2.4: More examples of video retrieval with nearest neighbour (same setting as Figure 8.1). Figure 2.4a is the NN retrieval with DPC pre-trained $f(.)$ on UCF101 (performance reported in Sec. 2.4.1). Figure 2.4b is the NN retrieval with ImageNet inflated $f(.)$. Retrieval is performed on UCF101 validation set.

We visualize the Nearest Neighbour (NN) of the video segments in the spatio-temporal feature space in Figure 2.4 and Figure 8.1. In detail, one video segment is randomly sampled from each video, then the spatio-temporal feature $z_i = f(x_i)$ is extracted and pooled into a vector. Then the feature vector is used to compute the cosine similarity score. In all figures, Figure 2.4a includes the video clips retrieved using our DPC model from self-supervised learning, note that the network does not receive any class label information during training. In comparison, Figure 2.4b uses the inflated ImageNet pre-trained weights.

It can be seen, that the ImageNet model is able to encode the scene semantics, e.g. human faces, crowds, but does not capture any semantics about the human actions. In contrast, our DPC model has actually learnt the video semantics *without* using any manual annotation, for instance, despite the background change in running, DPC can still correctly retrieve the video block.

2.5.2 Discussion

Why should the DPC model succeed in learning a representation suitable for action recognition, given the problem of a non-deterministic future? There are three reasons: First, the use of the softmax function and multi-way classification loss enables multi-modal, skewed, peaked or long tailed distributions; the model can

therefore handle the task of predicting the non-deterministic future. Second, by avoiding the shortcuts, the model has been prevented from learning simple smooth extrapolation of the embeddings; it is forced to learn semantic embeddings to succeed in its learning task. Third, in essence, DPC is trained by predicting future representations, and use them as a "query" to pick the correct "key" from lots of distractors. In order to succeed in this task, the model has to learn the shared semantics of the multiple possible future states, as this is the only way to always solve the multiple choice problem, no matter what future state appears along with the distractors. This common/shared representation is the invariance we are wishing for, i.e. igher level semantics. In other words, the representation of all these possible future states will be mapped to a space that their embeddings are close.

2.6 Conclusion

In this paper, we have introduced the Dense Predictive Coding (DPC) framework for self-supervised representation learning on videos, and outperformed the previous state-of-the-art by a large margin on the downstream tasks of action classification on UCF101 and HMDB51. As for future work, one straightforward extension of this idea is to employ different methods for aggregating the temporal information – instead of using a ConvGRU for temporal aggregation ($g(.)$ in the paper), other methods like masked CNN and attention based methods are also promising. In addition, empirical evidence shows that optical flow is able to boost the performance for action recognition significantly; it will be interesting to explore how optical flow can be trained jointly with DPC with self-supervised learning to further enhance the representation quality.

Acknowledgements

Funding for this research is provided by the Oxford-Google DeepMind Graduate Scholarship, and by the EPSRC Programme Grant Seebibyte EP/M013774/1.

Appendices

Appendices for this chapter can be found in the online version of the paper. [1]

Statement of authorship

A statement of authorship for this paper is provided in Appendix A.

[1] https://www.robots.ox.ac.uk/~vgg/research/DPC/dpc.html

Chapter 3

Memory-Augmented Dense Predictive Coding for Video Representation Learning

The paper has been accepted for publication as a spotlight presentation at the European Conference on Computer Vision (ECCV), 2020.

Memory-augmented Dense Predictive Coding for Video Representation Learning

Tengda Han Weidi Xie Andrew Zisserman

Visual Geometry Group, Department of Engineering Science, University of Oxford

{htd, weidi, az}@robots.ox.ac.uk

http://www.robots.ox.ac.uk/~vgg/research/DPC

Abstract

The objective of this paper is self-supervised learning from video, in particular for representations for action recognition. We make the following contributions: (i) We propose a new architecture and learning framework *Memory-augmented Dense Predictive Coding* (MemDPC) for the task. It is trained with a *predictive attention mechanism* over the set of *compressed memories*, such that any future states can always be constructed by a convex combination of the condensed representations, allowing to make multiple hypotheses efficiently. (ii) We investigate visual-only self-supervised video representation learning from RGB frames, or from unsupervised optical flow, or both. (iii) We thoroughly evaluate the quality of the learnt representation on four different downstream tasks: action recognition, video retrieval, learning with scarce annotations, and unintentional action classification. In all cases, we demonstrate state-of-the-art or comparable performance over other approaches with orders of magnitude fewer training data.

3.1 Introduction

Recent advances in self-supervised representation learning for images have yielded impressive results, e.g. [van den Oord et al. 2018; Hénaff et al. 2019; Hjelm et al. 2019; Yonglong Tian et al. 2020a; Ji et al. 2019; K. He et al. 2020; Zhuang et al. 2019; T. Chen et al. 2020], with performance matching or exceeding that of supervised representation learning on downstream tasks. However, in the case of videos, although there have been similar gains for *multi-modal* self-supervised representation learning, e.g. [Arandjelović and Zisserman 2017; Korbar et al. 2018; C. Sun et al. 2019; Alwassel et al. 2019; Piergiovanni et al. 2020; Miech et al. 2020], progress on learning *only* from the video stream (without additional audio or text streams) is lagging behind. The objective of this paper is to improve the performance of video only self-supervised learning.

Compared to still images, videos should be a more suitable source for self-supervised representation learning as they naturally provide various augmentation, such as object out of plane rotations and deformations. In addition, videos contain additional temporal information that can be used to disambiguate actions e.g. open vs. close. The temporal information can also act as a free supervisory signal to train a model to predict the future states from the past either passively by watching videos [Vondrick et al. 2016a; Lotter et al. 2017; Han et al. 2019] or actively in an interactive environment [Dosovitskiy and Koltun 2017], and thereby learn a video representation.

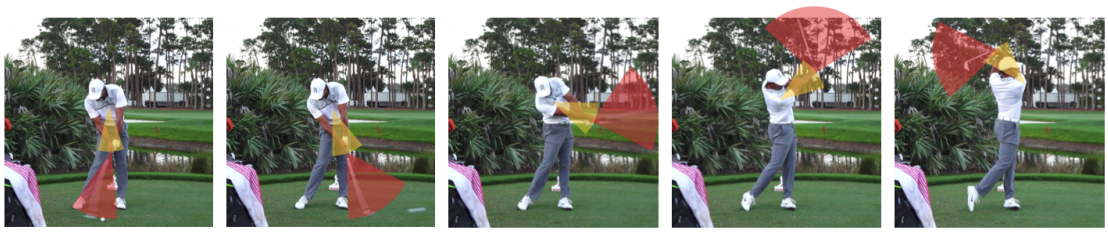


Figure 3.1: **Can you predict the next frame?** Future prediction naturally involves challenges from multiple hypotheses, e.g. the motion of each leaf, reflections on the water, hands and the golf club can be in many possible positions

Unfortunately, the exact future is indeterministic (a problem long discussed in the history of science, and known as "Laplace's Demon"). As shown in Figure 3.1, this problem is directly apparent in the stochastic variability of scenes, e.g. trying to predict the exact motion of each leaf on a tree when the wind blows, or the changing reflections on the water. More concretely, consider the action of 'playing

golf' – once the action starts, a future frame could have the hands and golf club in many possible positions, depending on the person who is playing. Learning visual representation by predicting the future therefore requires designing specific training schemes that simultaneously circumvents the unpredictable details in exact frames, and also handles multiple hypotheses and incomplete data – in particular only one possible future is exposed by the frames of one video.

Various approaches have been developed to deal with the multiple possible futures for an action. Vondrick *et al.* [Vondrick et al. 2016a] explicitly generates multiple hypotheses, and only the hypothesis that is closest to the true observation is chosen during optimization, however, this approach limits the number of possible future states. Another line of work [van den Oord et al. 2018; Han et al. 2019] circumvents this difficulty by using contrastive learning – the model is only asked to predict *one* future state that assigns higher similarity to the true observation than to any distractor observation (from different videos or from elsewhere in the same video). Recalling the 'playing golf' example, the embedding must capture the hand movement for this action, but not necessarily the precise position and velocity, only sufficiently to disambiguate future frames.

In this paper, we continue the idea of contrastive learning, but improve it by the addition of a *Compressive Memory*, which maps "lifelong" experience to a set of compressed memories and helps to better anticipate the future. We make the following four contributions: *First*, we propose a novel Memory-augmented Dense Predictive Coding (`MemDPC`) architecture. It is trained with a *predictive attention mechanism* over the set of *compressed memories*, such that any future states can always be constructed by a convex combination of the condensed representations, allowing it to make multiple hypotheses efficiently. *Second*, we investigate visual only self-supervised video representation learning from RGB frames, or from unsupervised optical flow, or both. *Third*, we argue that, in addition to the standard linear probes and fine-tuning [Richard Zhang et al. 2016; C. Sun et al. 2019], that have been used for evaluating representation from self-supervised learning, a non-linear probe should also be used, and demonstrate the difference that this probe makes. *Finally*, we evaluate the quality of learnt feature representation on four different downstream tasks: action recognition, learning under low-data regime (scarce annotations), video retrieval, and unintentional action classification; and demonstrate state-of-the-art performance over other approaches with similar

settings on *all* tasks.

3.2 Related Work

Self-supervised learning for images has undergone rapid progress in visual representation learning recently [van den Oord et al. 2018; Hénaff et al. 2019; Hjelm et al. 2019; Yonglong Tian et al. 2020a; Ji et al. 2019; K. He et al. 2020; Zhuang et al. 2019; T. Chen et al. 2020]. Generally speaking, the success can be attributed to one specific training paradigm, namely contrastive learning [Chopra et al. 2005; Gutmann and Hyvärinen 2010], i.e. contrast the positive and negative sample pairs.

Self-supervised learning for videos has explored various ideas to learn representations by exploiting spatio-temporal information [Isola et al. 2015; P. Agrawal et al. 2015; Jayaraman and Grauman 2015; X. Wang and Abhinav Gupta 2015; Misra et al. 2016; Vondrick et al. 2016a; Brabandere et al. 2016; Jayaraman and Grauman 2016; Lotter et al. 2017; Fernando et al. 2017; Lee et al. 2017; Wiles et al. 2018; Gan et al. 2018; Jakab et al. 2018; Jing and Yingli Tian 2018; Vondrick et al. 2018; Kim et al. 2019; D. Xu et al. 2019; Diba et al. 2019; Lai and W. Xie 2019; X. Wang et al. 2019; Alwassel et al. 2019; Lai et al. 2020]. Of more relevance here is the line of research using contrastive learning, e.g. [Arandjelović and Zisserman 2017; Arandjelović and Zisserman 2018; Korbar et al. 2018; Alwassel et al. 2019; Patrick et al. 2020; Piergiovanni et al. 2020] learn from visual-audio correspondence, [Miech et al. 2020] learns from video and narrations, and our previous work [Han et al. 2019] learns video representations by predicting future states.

Memory models have been considered as one of the fundamental building blocks towards intelligence. In the deep learning literature, two different lines of research have received extensive attention, one is to build networks that involve an internal memory which can be implicitly updated in a recurrent manner, e.g. LSTM [Hochreiter and Schmidhuber 1997] and GRU [Chung et al. 2014]. The other line of research focuses on augmenting feed-forward models with an explicit memory that can be read or written to with an attention-based procedure [K. Xu et al. 2015; Vaswani et al. 2017; Bahdanau et al. 2015; X. Wang et al. 2018; Devlin et al. 2018;

Kumar et al. 2016; Graves et al. 2014; Sukhbaatar et al. 2015]. In this work, our compressive memory falls in the latter line, i.e. an external memory module.

3.3 Methodology

The proposed Memory-augmented Dense Predictive Coding (MemDPC), is a conceptually simple model for learning a video representation with contrastive predictive coding. The key novelty is to augment the previous DPC model with a *Compressive Memory*. This provides a mechanism for handling the multiple future hypotheses required in learning due to the problem that only one possible future is exposed by a particular video.

The architecture is shown in Figure 3.2. As in the case of DPC, the video is partitioned into 8 blocks with 5 frames each, and an encoder network f generates an embedding z for each block. For inference, these embeddings are aggregated over time by a function g into a video level embedding c. During training, the future block embeddings $\hat{z}$ are predicted and used to select the true embedding in the dense predictive coding manner. In MemDPC, the prediction of $\hat{z}$ is from a convex combination of memory elements (rather from c directly as in DPC), and it is this restriction that also enables the network to handle multiple hypotheses, as will be explained below.

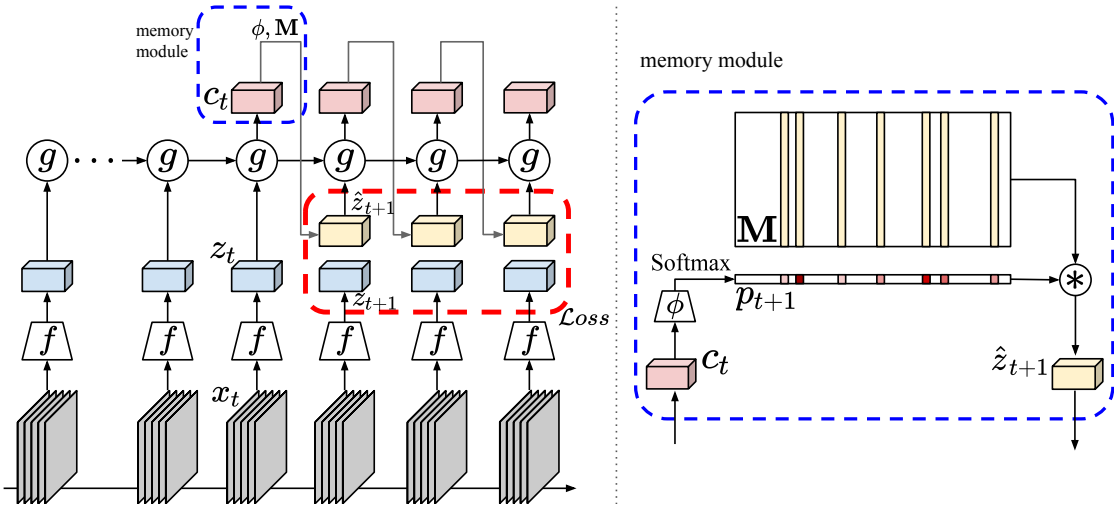


Figure 3.2: Architecture of the Memory-augmented Dense Predictive Coding (MemDPC). Note, the memory module is only used during the self-supervised training. The c_t embedding is used for downstream tasks

3.3.1 Memory-augmented Dense Predictive Coding (MemDPC)

Video Block Encoder. As shown in Figure 3.2, we partition the input video sequence into multiple blocks $x_1, ..., x_t, x_{t+1}, ...$, where each block is composed of multiple video frames. Then a shared feature extractor $f(.)$ (architecture details are given in the appendix) extracts the video features z_i from each video block x_i:

$$z_i = f(x_i) \tag{3.1}$$

Temporal Aggregation. After acquiring block representations, multiple block embeddings are aggregated to obtain a context feature c_t, summarizing the information over a longer temporal window. Specifically,

$$c_t = g(z_1, z_2, ..., z_t) \tag{3.2}$$

in our case, we simply adopt Recurrent Neural Networks (RNNs) for $g(.)$, but other auto-regressive model should also be feasible for temporal aggregation.

Compressive Memory. In order to enable efficient multi-hypotheses estimation, we augment the predictive models with an external common compressive memory. This external memory bank is shared for the entire dataset during training, and is accessed by a *predictive addressing mechanism* that infers a probability distribution over the memory entries, where each memory entry acts as a potential hypothesis.

In detail, the compressed memory bank is written $\mathbf{M} = [m_1, m_2, ..., m_k]^\top \in \mathbb{R}^{k \times C}$, where k is the memory size and C is the dimension of each compressed memory. During training, a predictive memory addressing mechanism (Eq. 3.3) is used to draw a hypothesis from the compressed memory, and the predicted future states $\hat{z}_{t+1}$ is then computed as the expectation of sampled hypotheses (Eq. 3.4):

$$p_{t+1} = \text{Softmax}\Big(\phi(c_t)\Big) \tag{3.3}$$

$$\hat{z}_{t+1} = \sum_{i=1}^{k} p_{(i,t+1)} \cdot m_i = p_{t+1}\mathbf{M} \tag{3.4}$$

where $p_{(i,t+1)} \in \mathbb{R}^k$ refers to the contribution of i-th memory slot for the future representation at time step t. A future prediction function $\phi(.)$ projects the context representation to $p_{(i,t+1)}$, in practice, $\phi(.)$ is learned with a multilayer perceptron.

The softmax operation is applied on the k dimension.

Multiple Hypotheses. The dot product of the predicted and desired future pairs can be rewritten as:

$$\psi(\hat{z}^\top, z) = \left(\sum_{i=1}^{k} p_i \cdot m_i^\top \right) z = \sum_{i=1}^{k} p_i \cdot \left(m_i^\top z \right) \tag{3.5}$$

where $m_i^\top z$ refers to the dot product (i.e. similarity) between a single memory slot and the feature states from the observation. The objective of $\phi(.)$ is to predict a probability distribution over k hypotheses in the memory bank, such that the expectation of $m_i^\top z$ for a positive pair is larger than that of negative pairs. Since the future is uncertain, the desired future feature z might be similar to one of the multiple hypotheses in the memory bank, say either m_p or m_q, for instance. To handle this uncertainty, the future prediction function $\phi(.)$ just needs to put a higher probability on both the p and q slots, such that Equation 3.5 is always large no matter which state the future is. In this way, the burden of modelling the future uncertainly is allocated to the memory bank $\mathbf{M}$ and future prediction function $\phi(.)$, thus the backbone encoder $f(.)$ and $g(.)$ can save capacity and capture the high-level action trajectory.

Memory Mechanism Discussion. Note, in contrast to the memory mechanism in Wu *et al.* [Z. Wu et al. 2018] and MoCo [K. He et al. 2020], which has the goal of storing more data samples to increase the number of negative samples during contrastive learning, our Compressive Memory has the goal of aiding learning by compressing all the potential hypotheses within the entire dataset, and allowing access through the predictive addressing mechanism. The memory mechanism shares similarity with NetVLAD [Arandjelović et al. 2016], which represents a feature distribution with "trainable centroids". However, in NetVLAD the goal is for compact and discriminative feature aggregation, and it encodes a weighted sum of *residuals* between feature vectors and the centroids. In contrast, our goal with $\phi(.)$ is to predict attention weights for the entries in the memory bank $\mathbf{M}$, in order to construct the future state as a convex combination these entries. The model can also sequentially predict further into the future with the *same* memory bank.

3.3.2 Contrastive Learning

Contrastive Learning generally refers to the training paradigm that forces the similarity scores of positive pairs to be higher than those of negatives.

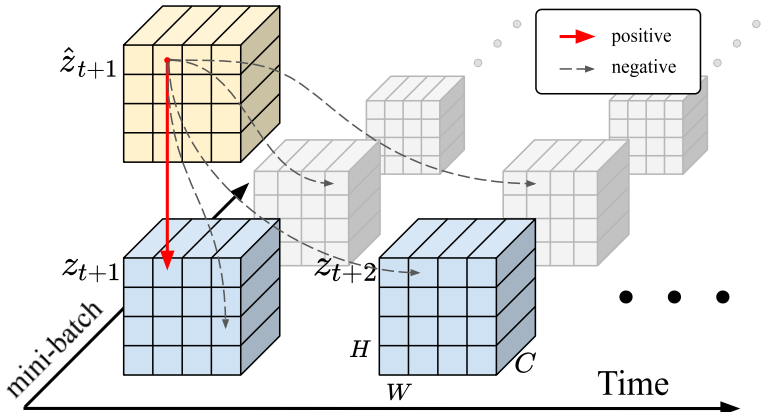


Figure 3.3: Details of the contrastive loss. Contrastive loss is computed densely, i.e. over both spatial and temporal dimension of the feature

Specifically, in `MemDPC`, we predict the future states recursively, ending up with a sequence of predicted features $\hat{z}_{t+1}, \hat{z}_{t+2}, \ldots, \hat{z}_{end}$ and the video feature from the true observations $z_{t+1}, z_{t+2}, \ldots, z_{end}$. As shown in Figure 3.3, each predicted $\hat{z}$ in practise is a dense feature map. To simplify the notation, we denote temporal index with i and denote other indexes including spatial index and batch-wise index as k, where batch-wise index means the index in the current mini-batch, $k \in \{(1,1,1),(1,1,2),...,(B,H,W)\}$. The objective function to minimize becomes:

$$\mathcal{L} = -\mathbb{E}\left[\sum_{i,k} \log \frac{e^{\psi(\hat{z}_{i,k}^{\top}, z_{i,k})}}{e^{\psi(\hat{z}_{i,k}^{\top}, z_{i,k})} + \sum_{(j,m)\neq(i,k)} e^{\psi(\hat{z}_{i,k}^{\top}, z_{j,m})}}\right] \tag{3.6}$$

where $\psi(\cdot)$ is acting as a *critic* function, in our case, we simply use dot product between the two vectors (we also experiment with L2-normalization, and find it gives similar performance on downstream tasks). Essentially, the objective function acts as a multi-way classifier, and the goal of optimization is to learn the video block encoder that assigns the highest values for $(\hat{z}_{i,k}, z_{i,k})$ i.e. higher similarity between the predicted future states and that from true observations originating from the same video and spatial-temporal aligned position.

3.3.3 Improving Performance by Extensions

As `MemDPC` is a general self-supervised learning framework, it can be combined with other 'modules' like two-stream networks and bi-directional RNN to improve the quality of the visual representations.

Two-stream Architecture. We represent dense optical flow as images by stacking the x and y displacement fields and another zero-valued array to make them 3-channel images. There is no need to modify the `MemDPC` framework, and it can be directly applied to optical flow inputs by simply replacing the input x_t from RGB frames to optical flow frames. We use late fusion like [Simonyan and Zisserman 2014; Feichtenhofer et al. 2016] to combine both streams.

Bi-directional `MemDPC`. From the perspective of human perception, where only the future is actively predicted, `MemDPC` is initially designed to be single-directional. However, when passively taking the videos as input, predicting backwards becomes feasible. Bi-directional `MemDPC` has a shared feature extractor $f(.)$ to extract the features $z_1, z_2, ..., z_t$, but has two identical aggregators $g^f(.)$ and $g^b(.)$ denoting forward and backward aggregation. They aggregate the bi-directional context features c_t^f and c_t^b. Then `MemDPC` predicts the past and the future features with the shared $\phi(.)$ and shared memory bank $\mathbf{M}$, and constructs contrastive losses for both directions, namely $\mathcal{L}^f$ and $\mathcal{L}^b$. The final loss is the average of the losses from both directions.

3.4 How to Evaluate Self-Supervised Learning?

The standard way to evaluate the quality of the learned representation is to assess the performance on downstream tasks using two protocols: (i) a linear probe – freezing the network and only train a linear head for the downstream task; or (ii) fine-tuning the entire network for the downstream task. For example, in (i) if the downstream task is classification, e.g. of UCF101, then a linear classifier is trained on top of the frozen base network. In (ii) the self-supervised training of the base network only provides the initialization. However, there is no particular reason why self-supervision should lead to features that are linearly separable, even if the

representation has encoded semantic information. Consequently, in addition to the two protocols mentioned above, we also evaluate the frozen features with non-linear probing, e.g. in the case of a classification downstream task, a non-linear MLP head is trained as the final classifier. In the experiments we evaluate the representation on four different downstream tasks.

Action Classification is a common evaluation task for self-supervised learning on videos and it allows us to compare against other methods. After self-supervised training, our `MemDPC` can be evaluated on this task under two settings: (i) linear and non-linear probing with a fixed network (here the entire backbone network, namely $f(.)$, $g(.)$); and (ii) fine-tuning the entire network end-to-end with supervised learning. For the embedding, as shown in Figure 3.4, we take the input video blocks $x_1, x_2, ..., x_t$ in the same way as `MemDPC` and extract the context feature c_t using the feature extractor $f(.)$ for each block and temporal aggregator $g(.)$; then we spatially pool the context feature c_t to obtain the embedding. We describe the training details in Section 3.5.3. The detailed experiment can be found in Section 3.5.4.

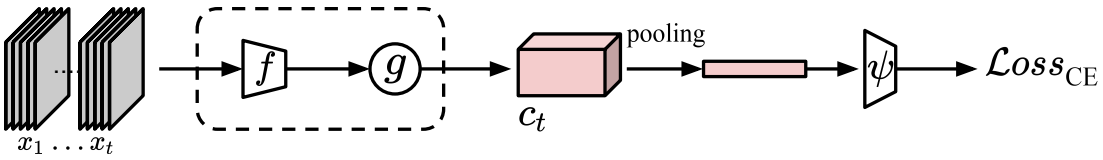


Figure 3.4: Architecture of the action classification framework

Data Efficiency and Generalizability are reflected by the effectiveness of the representation under a scarce-annotation regime. For this task, we take the `MemDPC` representation and finetune it for action classification task, but limit the model to only use 10%, 20% and 50% of the labelled training samples, then we report the accuracy on the same testing set. The classifier has the identical training pipeline as shown in Figure 3.4, and training details are explained in Section 3.5.3. The detailed experiment can be found in Section 3.5.5.

Video Action Retrieval directly evaluates the quality of the representation without any further training, aiming to provide a straightforward understanding on the quality of the learnt representation. Here, we use the simplest non-parametric clas-

sifier, i.e. k-nearest neighbours, to determine whether semantically similar actions are close in the high-dimensional space. Referring to Figure 3.4, for each video, we truncate it into blocks $x_1, x_2, ..., x_t$ and extract the context feature c_t with the $f(.)$ and $g(.)$ trained with `MemDPC`. We spatially pool c_t to get a context feature vector, which is directly used as a query vector for measuring the similarity with other videos in the dataset. The detailed experiment can be found in Section 3.5.6.

Unintentional Actions is a straightforward application for a predictive framework like `MemDPC`. We evaluate our representation on the task of unintentional event classification that is proposed in the recent Oops dataset [Epstein et al. 2020]. The core of unintentional events detection in video is a problem of anomaly detection. Usually, one of the predicted hypotheses tends to match true future relatively well for most of the videos. The discrepancy between them yields a measurement of future predictability, or 'surprise' level. A big surprise or a mismatched prediction can be used to locate the failing moment. In detail, we design the model as follows: first, we compute both the predicted feature $\hat{z}_i$ and the corresponding true feature z_i, and let a function $\xi(.)$ to measure their discrepancy. We train the model with two settings: (i) freezing the representation and only train the classifier $\xi(.)$; (ii) finetuning the entire network. The detailed structure for the classification task can be found in Section 3.5.7.

3.5 Experiments

3.5.1 Datasets

For the self-supervised training, two video action recognition datasets are used, but labels are dropped during training: *UCF101* [Soomro et al. 2012], containing 13k videos spanning over 101 human actions; and Kinetics400 (*K400*) [Kay et al. 2017] with 306k 10-second video clips covering 400 human actions. For the downstream tasks we also use UCF101, and additionally we use: *HMDB51* [Hildegard Kuehne et al. 2011] containing 7k videos spanning over 51 human actions; and *Oops* [Epstein et al. 2020] containing 20k videos of daily human activities with unexpected failed moments, among them 14k videos have the time stamps of the failed moments manually labelled.

3.5.2 Self-Supervised Training

In our experiment, we use a (2+3D)-ResNet, following [Feichtenhofer et al. 2019a; Han et al. 2019], as the encoder $f(.)$, where the first two residual blocks res2 and res3 have 2D convolutional kernels, and only res4 and res5 have 3D kernels. Specifically, (2+3D)-ResNet18 and (2+3D)-ResNet34 are used in our experiments, denoted as R18 and R34 below. For the temporal aggregation, $g(\cdot)$, we use an one-layer GRU with kernel size 1×1, with the weights shared among all spatial positions on the feature map. The future prediction function, $\phi(.)$, is a two-layer convolutional network. We choose the size of the memory bank $\mathbf{M}$ to be 1024 based on experiments in Table 3.1. Network architecture are given in the appendix.

For the data, raw videos are decoded at a frame rate 24-30 fps, and each data sample consists of 40 consecutive frames, sampled with a temporal stride of 3 from the raw video. As input to `MemDPC`, they are divided into 8 video clips – so that each encoder $f(.)$ inputs 5 frames, covering around 0.5 seconds, and the 40 frames around 4 seconds. For optical flow, in order to eliminate extra supervisory signals in the self-supervised training stage, we use the **un-supervised** TV-L1 algorithm [Zach et al. 2007], and follow the same pre-processing procedures as [João Carreira and Zisserman 2017], i.e. truncating large motions with more than ± 20 in both channels, appending a third 0s channel, and transforming the values from $[-20, 20]$ to $[0, 255]$. For data augmentation, we apply clip-wise random crop and horizontal flip, and frame-wise color jittering and random greyscale, for both the RGB and optical flow streams. We experiment with both 128×128 and 224×224 input resolution. The original video resolution is 256×256 and it is firstly cropped to 224×224 then rescaled if needed. Self-supervised training uses the Adam [Kingma and Ba 2014a] optimizer with initial learning rate 10^{-3}. The learning rate is decayed once to 10^{-4} when the validation loss plateaus. We use a batch size of 16 samples per GPU.

3.5.3 Supervised Classification

For all action classification downstream tasks, the input follows the same frame sampling procedure as when the model is trained with self-supervised learning, and then we train the classifier with cross-entropy loss as shown in Figure 3.4. A dropout of 0.9 is applied on the final layer. For data augmentation, we use

clip-wise random crop, random horizontal flip, and random color jittering. The classifier is trained with Adam with a 10^{-3} initial learning rate, and decayed once to 10^{-4} when the validation loss plateaus. During testing, we follow the standard pipeline, i.e. ten-crop (center and four corner crops, w/o horizontal flip), take the same sequence length as training from the video, and average the prediction from the sampling temporal moving window.

3.5.4 Evaluation: Action Classification

We conduct two sets of experiments: (i) ablation studies on the effectiveness of the different modules in the `MemDPC`, by self-supervised learning on UCF101, (ii) to compare with other state-of-the-art approaches, we run `MemDPC` on K400 with self-supervised learning. For both settings, the representation quality is evaluated on UCF101 and HMDB51 with linear probing, non-linear probing, and end-to-end finetuning.

Ablations on UCF101.

In this section, we conduct extensive experiments to validate the effectiveness of compressive memory, bidirectional aggregation, and self-supervised learning on optical flow. Note that, in each experiment, we keep the settings identical, and only vary one variable at a time.

As shown in Table 3.1, the following phenomena can be observed: *First*, comparing experiment $\mathcal{C}2$ against $\mathcal{B}1$ (68.2 vs. 61.8), networks initialized with self-supervised `MemDPC` clearly present better generalization than a randomly initialized network; *Second*, comparing with a strong baseline ($\mathcal{A}$), the proposed compressive memory boost the learnt representation by around 5% (68.2 vs. 63.6), and the optimal memory size for UCF101 is 1024; *Third*, `MemDPC` acts as a general learning framework that can also help to boost the generalizability of motion representations, a 7.3% boost can be seen from $\mathcal{D}1$ vs. $\mathcal{B}2$ (81.9 vs. 74.6); *Fourth*, the bidirectional aggregation provides a small boost to the accuracy by about 1% ($\mathcal{E}1$ vs. $\mathcal{C}2$, $\mathcal{E}2$ vs. $\mathcal{D}1$, $\mathcal{E}3$ vs. $\mathcal{D}2$). *Lastly*, after fusing both streams, $\mathcal{D}2$ achieves 84% classification accuracy, confirming our claim that self-supervised learning with only the video stream (without additional audio or text streams) can also end up with strong action recognition models.

#	Network	Dataset	Input	Self-Sup. Resolution	Memory size	Sup. (top1) UCF101(ft)
$\mathcal{A}$	R18	UCF101	RGB	128 × 128	- (DPC [Han et al. 2019])	63.6
$\mathcal{B}1$	R18	-(rand. init.)	RGB	128 × 128	-	61.8
$\mathcal{B}2$	R18	-(rand. init.)	Flow	128 × 128	-	74.6
$\mathcal{B}3$	R18×2	-(rand. init.)	RGB+F	128 × 128	-	78.7
$\mathcal{C}1$	R18	UCF101	RGB	128 × 128	512	65.3
$\mathcal{C}2$	R18	UCF101	RGB	128 × 128	1024	68.2
$\mathcal{C}3$	R18	UCF101	RGB	128 × 128	2048	68.0
$\mathcal{D}1$	R18	UCF101	Flow	128 × 128	1024	81.9
$\mathcal{D}2$	R18×2	UCF101	RGB+F	128 × 128	1024	84.0
$\mathcal{E}1$	R18-bd	UCF101	RGB	128 × 128	1024	69.2
$\mathcal{E}2$	R18-bd	UCF101	Flow	128 × 128	1024	82.3
$\mathcal{E}3$	R18-bd×2	UCF101	RGB+F	128 × 128	1024	84.3

Table 3.1: Ablation studies. We train `MemDPC` on UCF101 and evaluate on UCF101 action classification by finetuning the entire network. We group rows for clarity: $\mathcal{A}$ is the reimplementation of DPC, $\mathcal{B}$ are random initialization baselines, $\mathcal{C}$ for different memory size, $\mathcal{D}$ incorporates optical flow, $\mathcal{E}$ incorporates a bi-directional RNN

Comparison with others.

In this section, we train `MemDPC` on K400 and evaluate the action classification performance on UCF101 and HMDB51. Specifically, we evaluate three settings: (1) finetuning the entire network (denoted as Freeze=✗); (2) freeze the backbone and only train a linear classifier, i.e. linear probe (denoted as Freeze=✓); (3) freeze the backbone and only train a **non-linear** classifier, i.e. non-linear probe (denoted as 'n.l.').

As shown in Table 8.5, for the same amount of data (K400) and visual-only input, `MemDPC` surpasses all previous state-of-the-art self-supervised methods on both UCF101 and HMDB51 (although there exist small differences in architecture, e.g. for 3DRotNet, ST-Puzzle, DPC, SpeedNet). When freezing the representation, it can be seen that a non-linear probe gives better results than a linear probe, and in practice a non-linear classifier is still very cheap to train.

Other self-supervised training methods on the same benchmarks are not directly comparable, even ignoring the architecture differences, due to the duration of videos used or to the number of modalities used. For example, CBT [C. Sun et al. 2019] uses a longer version of K600 (referred to as K600+ in the table), the size is about 9 times that of the standard K400 that we use, and CBT requires RotNet [Jing and Yingli Tian 2018] initialization while `MemDPC` can be trained from scratch. Nevertheless, our performance exceeds that of CBT. Other works use additional modalities for pre-text tasks like audio [Korbar et al. 2018; Alwassel et al. 2019; Patrick et al. 2020; Piergiovanni et al. 2020], or narrations [Miech et al.

Method	Date	Dataset (duration)	Res.	Arch.	Depth	Modality	Frozen	UCF	HMDB
CBT [C. Sun et al. 2019]	2019	K600+ (273d)	112	S3D	23	V	✓	54.0	29.5
MIL-NCE [Miech et al. 2020]	2020	HTM (15y)	224	S3D	23	V+T	✓	82.7	53.1
MIL-NCE [Miech et al. 2020]	2020	HTM (15y)	224	I3D	22	V+T	✓	83.4	54.8
XDC [Alwassel et al. 2019]	2019	IG65M (21y)	224	R(2+1)D	26	V+A	✓	85.3	56.0
ELO [Piergiovanni et al. 2020]	2020	Youtube8M- (8y)	224	R(2+1)D	65	V+A	✓	–	64.5
MemDPC†		K400 (28d)	224	R-2D3D	33	V	✓	54.1	30.5
MemDPC†		K400 (28d)	224	R-2D3D	33	V	✓n.l.	58.5	33.6
OPN [Lee et al. 2017]	2017	UCF (1d)	227	VGG	14	V	✗	59.6	23.8
3D-RotNet [Jing and Yingli Tian 2018]	2018	K400 (28d)	112	R3D	17	V	✗	62.9	33.7
ST-Puzzle [Kim et al. 2019]	2019	K400 (28d)	224	R3D	17	V	✗	63.9	33.7
VCOP [D. Xu et al. 2019]	2019	UCF (1d)	112	R(2+1)D	26	V	✗	72.4	30.9
DPC [Han et al. 2019]	2019	K400 (28d)	224	R-2D3D	33	V	✗	75.7	35.7
CBT [C. Sun et al. 2019]	2019	K600+ (273d)	112	S3D	23	V	✗	79.5	44.6
DynamoNet [Diba et al. 2019]	2019	Youtube8M-1 (1.9y)	112	STCNet	133	V	✗	88.1	59.9
SpeedNet [Benaim et al. 2020]	2020	K400 (28d)	224	S3D-G	23	V	✗	81.1	48.8
AVTS [Korbar et al. 2018]	2018	K400 (28d)	224	I3D	22	V+A	✗	83.7	53.0
AVTS [Korbar et al. 2018]	2018	AudioSet (240d)	224	MC3	17	V+A	✗	89.0	61.6
XDC [Alwassel et al. 2019]	2019	K400 (28d)	224	R(2+1)D	26	V+A	✗	84.2	47.1
XDC [Alwassel et al. 2019]	2019	IG65M (21y)	224	R(2+1)D	26	V+A	✗	94.2	67.4
GDT [Patrick et al. 2020]	2020	K400 (28d)	112	R(2+1)D	26	V+A	✗	88.7	57.8
MIL-NCE [Miech et al. 2020]	2020	HTM (15y)	224	S3D-G	23	V+T	✗	91.3	61.0
ELO [Piergiovanni et al. 2020]	2020	Youtube8M-2 (13y)	224	R(2+1)D	65	V+A	✗	93.8	67.4
MemDPC		K400 (28d)	224	R-2D3D	33	V	✗	78.1	41.2
MemDPC†		K400 (28d)	224	R-2D3D	33	V	✗	86.1	54.5
Supervised [Hara et al. 2018]		K400 (28d)	224	R3D	33	V	✗	87.7	59.1

Table 3.2: Comparison with state-of-the-art approaches. In the left columns, we show the pre-training setting, e.g. dataset, resolution, architectures with encoder depth, modality. In the right columns, the top-1 accuracy is reported on the downstream action classification task for different datasets, e.g. UCF, HMDB, K400. The dataset parenthesis shows the total video duration in time (**d** for day, **y** for year). 'Frozen ✗' means the network is end-to-end finetuned from the pretrained representation, shown in the top half of the table; 'Frozen ✓' means the pretrained representation is fixed and classified with a linear layer, 'n.l.' denotes a non-linear classifier. For input, 'V' refers to visual only (colored with blue), 'A' is audio, 'T' is text narration. MemDPC models with † refer to the two-stream networks, where the predictions from RGB and Flow networks are averaged

2020], and train on larger datasets. Despite these disadvantages, we demonstrate that `MemDPC` trained with only visual inputs, can achieve competitive results on the finetuning protocol.

3.5.5 Evaluation: Data Efficiency

In Figure 3.5, we show the data efficiency of `MemDPC` on both RGB input and optical flow with action recognition on the UCF101 dataset. As we reduce the labelled training samples, action classifier trained on `MemDPC` representation generalize significantly better than the classifier trained from scratch. Also, to match the performance of a random initialized classifier trained on 100% labelled data, a classifier trained on `MemDPC` initialization only requires less than 50% labelled data for both RGB and optical flow input.

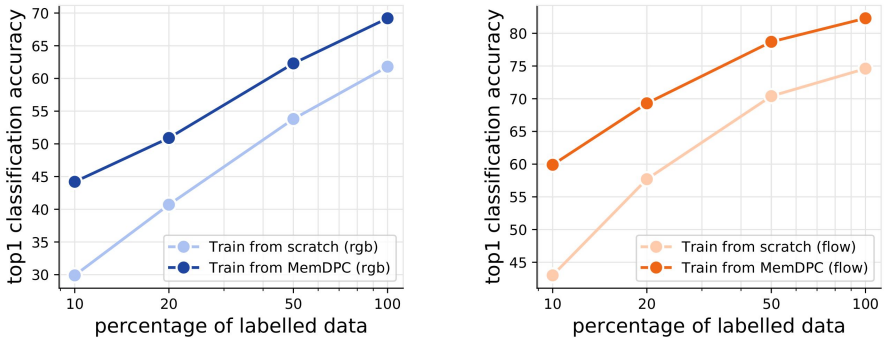


Figure 3.5: Data efficiency of `MemDPC` representations. Left is RGB input and right is optical flow input. The `MemDPC` is trained on UCF101 and it is evaluated on action classification (finetuning protocol) on UCF101 with a reduced number of labels

3.5.6 Evaluation: Video Retrieval

In this protocol, we evaluate our representation with nearest-neighbour video retrieval, features are extracted from the model, which is only trained with self-supervised learning, no further finetuning is allowed.

Experiments are shown on two datasets: UCF101 and HMDB51. For both datasets, within the training set or within the testing set, multiple clips could be from the same source video, hence they are visually similar and make the retrieval task trivial. We follow the practice of [D. Xu et al. 2019; D. Luo et al. 2020], and use each clip in the test set to query the k nearest clips in the training set.

For each clip, we sample multiple 8 video blocks with a sliding window, and extract the context representation c_t for each window. We spatial-pool each c_t and take the average over all the windows. For distance measurement, we use cosine distance. We report Recall at k (R@k) as the evaluation metric. That is, as long as one clip of the same class is retrieved in the top k nearest neighbours, a correct retrieval is counted.

Method	Date	Dataset	UCF				HMDB			
			R@1	R@5	R@10	R@20	R@1	R@5	R@10	R@20
Jigsaw [Noroozi and Favaro 2016]	2016	UCF	19.7	28.5	33.5	40.0	-	-	-	-
OPN [Lee et al. 2017]	2017	UCF	19.9	28.7	34.0	40.6	-	-	-	-
Buchler [Büchler et al. 2018]	2018	UCF	25.7	36.2	42.2	49.2	-	-	-	-
VCOP [D. Xu et al. 2019]	2019	UCF	14.1	30.3	40.4	51.1	7.6	22.9	34.4	48.8
VCP [D. Luo et al. 2020]	2020	UCF	18.6	33.6	42.5	53.5	7.6	24.4	36.3	53.6
SpeedNet [Benaim et al. 2020]	2020	K400	13.0	28.1	37.5	49.5	-	-	-	-
`MemDPC`-RGB		UCF	20.2	**40.4**	**52.4**	**64.7**	**7.7**	**25.7**	**40.6**	**57.7**
`MemDPC`-Flow		UCF	**40.2**	**63.2**	**71.9**	**78.6**	**15.6**	**37.6**	**52.0**	**65.3**

Table 3.3: Comparison with others on Nearest-Neighbour video retrieval on UCF101 and HMDB51. Testing set clips are used to retrieve training set videos and R@k is reported, where $k \in [1, 5, 10, 20]$. Note that all the models reported were only pretrained on UCF101 with self-supervised learning except SpeedNet

In Table 4.3, we show the retrieval performance on UCF101 and HMDB51. Note that the MemDPC benchmarked here is only trained on UCF101, the same as [D. Xu et al. 2019; D. Luo et al. 2020]. For fair comparison, MemDPC in this experiment uses a R18 backbone, which has the same depth but less parameters than the 3D-ResNet used in [D. Xu et al. 2019; D. Luo et al. 2020]. With RGB inputs, our MemDPC gets state-of-the-art performance on all the metrics except R@1 in UCF101, where the method from Buchler *et al.* [Büchler et al. 2018] specializes well on R@1. While for Flow inputs, MemDPC significantly outperforms all previous methods by a large margin. We also qualitatively show video retrieval results in the Appendix.

3.5.7 Evaluation: Unintentional Actions

We evaluate MemDPC on the Oops dataset on unintentional action classification. In Oops, there is one failure moment in the middle of each video. When cutting the video into short clips, the clip overlapping the failure moment is defined as a 'transitioning' action, the clips before are 'intentional' actions, and the clips afterwards are 'unintentional' actions. The core task is therefore to classify each short video clip into one of three categories,

In this experiment, we use a R18 based MemDPC model that takes 128×128 resolution video frames as input. After MemDPC is trained on K400 and the Oops training set videos with self-supervised learning, we further train it for unintentional action classification with a linear probe, and end-to-end finetuning (as shown in Table 3.4). The training details are given in the appendix. State-of-the-art performance is demonstrated by our MemDPC on this unintentional action classification task, even outperforming the model pretrained on K700 with full supervision with finetuning.

Task	Method	Backbone	Freeze	Finetune
Classification	K700 Supervision	3D-ResNet18	**53.6**	64.0
	Video Speed [Epstein et al. 2020]	3D-ResNet18	53.4	61.6
	MemDPC	R18	53.0	**64.4**

Table 3.4: MemDPC on unintentional action classification tasks. Note that our backbone 2+3D-ResNet18 has the same depth as 3D-ResNet18 used in [Epstein et al. 2020] but with fewer parameters. MemDPC model is trained on K400 and the OOPS training set without using labels, and the network is then finetuned with supervision from the OOPS training set

3.6 Conclusion

In this paper, we propose a new architecture and learning framework (`MemDPC`) for self-supervised learning from video, in particular for representations for action recognition. With the novel compressive memory, the model can efficiently handle the nature of multiple hypotheses in the self-supervised predictive learning procedure. In order to thoroughly evaluate the quality of the learnt representation, we conduct experiments on four different downstream tasks, namely action recognition, video retrieval, learning with scarce annotations, and unintentional action classification. In all cases, we demonstrate state-of-the-art or competitive performance over other approaches that use orders of magnitude more training data. Above all, for the first time, we show that it is possible to learn high-quality video representations with self-supervised learning, from the visual stream alone (without additional audio or text streams).

Acknowledgements.

Funding for this research is provided by a Google-DeepMind Graduate Scholarship, and by the EPSRC Programme Grant Seebibyte EP/M013774/1. We would like to thank João F. Henriques, Samuel Albanie and Triantafyllos Afouras for helpful discussions.

Appendices

Appendices for this chapter can be found in the online version of the paper. [1]

Statement of authorship

A statement of authorship for this paper is provided in Appendix A.

[1] https://www.robots.ox.ac.uk/~vgg/research/DPC/

Chapter 4

Self-supervised Co-training for Video Representation Learning

The paper has been accepted for publication at the Neural Information Processing Systems (NeurIPS), 2020.

Self-supervised Co-training for Video Representation Learning

Tengda Han Weidi Xie Andrew Zisserman
Visual Geometry Group, Department of Engineering Science,
University of Oxford
{htd, weidi, az}@robots.ox.ac.uk

http://www.robots.ox.ac.uk/~vgg/research/CoCLR/.

Abstract

The objective of this paper is visual-only self-supervised video representation learning. We make the following contributions: (i) we investigate the benefit of adding semantic-class positives to instance-based Info Noise Contrastive Estimation (InfoNCE) training, showing that this form of supervised contrastive learning leads to a clear improvement in performance; (ii) we propose a novel *self-supervised co-training* scheme to improve the popular infoNCE loss, exploiting the complementary information from different views, RGB streams and optical flow, of the same data source by using one view to obtain positive class samples for the other; (iii) we thoroughly evaluate the quality of the learnt representation on two different downstream tasks: action recognition and video retrieval. In both cases, the proposed approach demonstrates state-of-the-art or comparable performance with other self-supervised approaches, whilst being significantly more efficient to train, i.e. requiring far less training data to achieve similar performance.

4.1 Introduction

The recent progress in self-supervised representation learning for images and videos has demonstrated the benefits of using a discriminative contrastive loss on data samples [van den Oord et al. 2018; Hénaff et al. 2019; K. He et al. 2020; Misra and van der Maaten 2020; T. Chen et al. 2020; X. Chen et al. 2020], such as NCE [Gutmann and Hyvärinen 2010; Jozefowicz et al. 2016]. Given a data sample, the objective is to discriminate its transformed version against other samples in the dataset. The transformations can be artificial, such as those used in data augmentation [T. Chen et al. 2020], or natural, such as those arising in videos from temporal segments within the same clip. In essence, these pretext tasks focus on *instance discrimination*: each data sample is treated as a 'class', and the objective is to discriminate its own augmented version from a large number of other data samples or their augmented versions. Representations learned by instance discrimination in this manner have demonstrated extremely high performance on downstream tasks, often comparable to that achieved by supervised training [K. He et al. 2020; T. Chen et al. 2020].

In this paper, we target self-supervised video representation learning, and ask the question: **is instance discrimination making the best use of data?** We show that the answer is *no*, in two respects:

First, we show that hard positives are being neglected in the self-supervised training, and that if these hard positives are included then the quality of learnt representation improves significantly. To investigate this, we conduct an *oracle* experiment where positive samples are incorporated into the instance-based training process based on the semantic class label. A clear performance gap is observed between the pure instance-based learning (termed *InfoNCE* [van den Oord et al. 2018]) and the oracle version (termed *UberNCE*). Note that the oracle is a form of supervised contrastive learning, as it encourages linear separability of the feature representation according to the class labels. In our experiments, training with UberNCE actually outperforms the supervised model trained with cross-entropy, a phenomenon that is also observed in a concurrent work [Khosla et al. 2020] for image classification.

Second, we propose a self-supervised co-training method, called *CoCLR*, standing for 'Co-training Contrastive Learning of visual Representation', with the goal

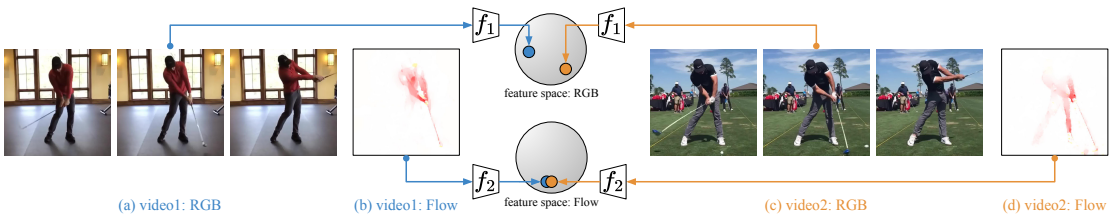


Figure 4.1: Two video clips of a golf-swing action and their corresponding optical flows. In this example, the flow patterns are very similar across different video instances despite significant variations in RGB space. This observation motivates the idea of co-training, which aims to gradually enhance the representation power of both networks, $f_1(\cdot)$ and $f_2(\cdot)$, by mining hard positives from one another.

of mining positive samples by using other complementary *views* of the data, i.e. replacing the role of the *oracle*. We pick RGB video frames and optical flow as the two views from hereon. As illustrated in Figure 8.1, positives obtained from flow can be used to 'bridge the gap' between the RGB video clips instances. In turn, positives obtained from RGB video clips can link optical flow clips of the same action. The outcome of training with the CoCLR algorithm is a representation that significantly surpasses the performance obtained by the instance-based training with InfoNCE, and approaches the performance of the *oracle* training with UberNCE.

To be clear, we are not proposing a new loss function or pretext task here, but instead we target the training regime by improving the sampling process in the contrastive learning of visual representation, i.e. constructing positive pairs beyond instances. There are two benefits: first, (hard) positive examples of the same class (e.g. the golf-swing action shown in Figure 8.1) are used in training; second, these positive samples are removed from the instance level negatives – where they would have been treated as false negatives for the action class. Our primary interest in this paper is to improve the representation of both the RGB and Flow networks, using the complementary information provided by the other view. For inference, we may choose to use only the RGB network or the Flow network, or both, depending on the applications and efficiency requirements.

To summarize, we investigate visual-only self-supervised video representation learning from RGB frames, or from unsupervised optical flow, or from both, and make the following contributions: (i) we show that an oracle with access to semantic class labels improves the performance of instance-based contrastive learning; (ii) we propose a novel self-supervised co-training scheme, CoCLR, to improve the

training regime of the popular InfoNCE, exploiting the complementary information from different views of the same data source; and (iii) we thoroughly evaluate the quality of the learnt representation on two downstream tasks, video action recognition and retrieval, on UCF101 and HMDB51. In all cases, we demonstrate state-of-the-art or comparable performance over other self-supervised approaches, while being significantly more efficient, i.e. less data is required for self-supervised pre-training.

Our observations of using a second complementary view to bridge the RGB gap between positive instances from the same class is applied in this paper to optical flow. However, the idea is generally applicable for other complementary views: for videos, audio or text narrations can play a similar role to optical flow; whilst for still images, the multiple views can be formed by passing images through different filters. We return to this point in Section 8.7.

4.2 Related work

Visual-only Self-supervised Learning. Self-supervised visual representation learning has recently witnessed rapid progress in image classification. Early work in this area defined proxy tasks explicitly, for example, colorization, inpainting, and jigsaw solving [Doersch et al. 2015; Y. Zhang et al. 2016; Pathak et al. 2016; Doersch and Zisserman 2017]. More recent approaches jointly optimize clustering and representation learning [Caron et al. 2018; Caron et al. 2019; Yuki Markus Asano et al. 2020] or learn visual representation by discriminating instances from each other through contrastive learning [van den Oord et al. 2018; Hénaff et al. 2019; Hjelm et al. 2019; Yonglong Tian et al. 2020a; Ji et al. 2019; Zhuang et al. 2019; K. He et al. 2020; Misra and van der Maaten 2020; T. Chen et al. 2020]. Videos offer additional opportunities for learning representations, beyond those of images, by exploiting spatio-temporal information, for example, by ordering frames or clips [Misra et al. 2016; Lee et al. 2017; Fernando et al. 2017; D. Xu et al. 2019; Wei et al. 2018], motion [P. Agrawal et al. 2015; Jayaraman and Grauman 2015; Diba et al. 2019], co-occurrence [Isola et al. 2015], jigsaw [Kim et al. 2019], rotation [Jing and Yingli Tian 2018], speed prediction [Epstein et al. 2020; Benaim et al. 2020; Jiangliu Wang et al. 2020], future prediction [Vondrick et al. 2016a; Han et al. 2019; Han et al. 2020a], or by temporal coherence [Vondrick et al. 2018;

Lai and W. Xie 2019; X. Wang et al. 2019; Lai et al. 2020].

Multi-modal Self-supervised Learning. This research area focuses on leveraging the interplay of different modalities, for instance, contrastive loss is used to learn the correspondence between frames and audio [Arandjelović and Zisserman 2017; Arandjelović and Zisserman 2018; Korbar et al. 2018; Alwassel et al. 2019; Patrick et al. 2020; Piergiovanni et al. 2020], or video and narrations [Miech et al. 2020]; or, alternatively, an iterative clustering and re-labelling approach for video and audio has been used in [Alwassel et al. 2019].

Co-training Paired Networks. Co-training [Blum and Mitchell 1998] refers to a semi-supervised learning technique that assumes each example to be described by multiple views that provide different and complementary information about the instance. Co-training first learns a separate classifier for each view using any labelled examples, and the most confident predictions of each classifier on the unlabelled data are then used to iteratively construct additional labelled training data. Though note in our case that we have *no* labelled samples. More generally, the idea of having two networks interact and co-train also appears in other areas of machine learning, e.g. Generative Adversarial Networks (GANs) [Goodfellow et al. 2014], and Actor-Critic Reinforcement Learning [Sutton et al. 2000].

Video Action Recognition. This research area has gone through a rapid development in recent years, from the two-stream networks [Simonyan and Zisserman 2014; Feichtenhofer et al. 2016; Diba et al. 2019; H. Zhao et al. 2019] to the more recent single stream RGB networks [Tran et al. 2018; S. Xie et al. 2018; Feichtenhofer et al. 2019b; Feichtenhofer 2020], and the action classification performance has steadily improved. In particular, the use of distillation [Stroud et al. 2018; Diba et al. 2019], where the flow-stream network is used to teach the RGB-stream network, at a high level, is related to the goal in this work.

4.3 InfoNCE, UberNCE and CoCLR

We first review instance discrimination based self-supervised learning with InfoNCE, as used by [T. Chen et al. 2020], and introduce an *oracle* extension where positive samples are incorporated into the instance-based training process based on the semantic class label. Then in Section 4.3.2, we introduce the key idea of mining informative positive pairs using multiview co-training, and describe our

algorithm for employing it, which enables InfoNCE to extend beyond instance discrimination.

4.3.1 Learning with InfoNCE and UberNCE

InfoNCE. Given a dataset with N raw video clips, e.g. $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$, the objective for self-supervised video representation learning is to obtain a function $f(\cdot)$ that can be effectively used to encode the video clips for various downsteam tasks, e.g. action recognition, retrieval, *etc*.

Assume there is an augmentation function $\psi(\cdot; a)$, where a is sampled from a set of pre-defined data augmentation transformations A, that is applied to $\mathcal{D}$. For a particular sample x_i, the positive set $\mathcal{P}_i$ and the negative set $\mathcal{N}_i$ are defined as: $\mathcal{P}_i = \{\psi(x_i; a) | a \sim A\}$, and $\mathcal{N}_i = \{\psi(x_n; a) | \forall n \neq i, a \sim A\}$. Given $z_i = f(\psi(x_i; \cdot))$, then the InfoNCE loss is:

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}\left[\log \frac{\exp\left(z_i \cdot z_p/\tau\right)}{\exp\left(z_i \cdot z_p/\tau\right) + \sum_{n \in \mathcal{N}_i} \exp\left(z_i \cdot z_n/\tau\right)}\right] \tag{4.1}$$

where $z_i \cdot z_p$ refers to the dot product between two vectors. In essence, the objective for optimization can be seen as instance discrimination, i.e. emitting higher similarity scores between the augmented views of the *same* instance than with augmented views from *other* instances.

UberNCE. Assume we have a dataset with annotations, $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where y_i is the class label for clip x_i, and an *oracle* that has access to these annotations. We search for a function $f(\cdot)$, by optimizing an identical InfoNCE to Eq. 4.1, *except* that for each sample x_i, the positive set $\mathcal{P}_i$ and the negative set $\mathcal{N}_i$ can now include samples with same semantic labels, in addition to the augmentations, i.e. $\mathcal{P}_i = \{\psi(x_i; a), x_p | y_p = y_i \text{ and } p \neq i, \forall p \in [1, N], a \sim A\}$, $\mathcal{N}_i = \{\psi(x_n; a), x_n | y_n \neq y_i, \forall n \in [1, N], a \sim A\}$.

As an example, given an input video clip of a 'running' action, the positive set contains its own augmented version and all other 'running' video clips in the dataset, and the negative set consists all video clips from other action classes.

As will be demonstrated in Section 4.4.4, we evaluate the representation on a linear probe protocol, and observe a significant performance gap between training on InfoNCE and UberNCE, confirming that **instance discrimination is not**

making the best use of data. Clearly, the choice of sampling more informative *positives* i.e. treating semantically related clips as positive pairs (and thereby naturally eliminating *false negatives*), plays a vital role in such representation learning, as this is the *only* difference between InfoNCE and UberNCE.

4.3.2 Self-supervised CoCLR

As an extension of the previous notation, given a video clip x_i, we now consider two different views, $\mathbf{x}_i = \{x_{1i}, x_{2i}\}$, where in this paper, x_{1i} and x_{2i} refer to RGB frames and their *unsupervised* optical flows respectively. The objective of self-supervised video representation learning is to learn the functions $f_1(\cdot)$ and $f_2(\cdot)$, where $z_{1i} = f_1(x_{1i})$ and $z_{2i} = f_2(x_{2i})$ refer to the representations of the RGB stream and optical flow, that can be effectively used for performing various downstream tasks.

The key idea, and how the method differs from InfoNCE and UberNCE, is in the construction of the positive set ($\mathcal{P}_i$) and negative set ($\mathcal{N}_i$) for the sample x_i. Intuitively, positives that are very hard to 'discover' in the RGB stream can often be 'easily' determined in the optical flow stream. For instance, under static camera settings, flow patterns from a particular action, such as golf swing, can be very similar across instances despite significant background variations that dominate the RGB representation (as shown in Figure 8.1). Such similarities can be discovered even with a partially trained optical flow network. This observation enables two models, one for RGB and the other for flow, to be co-trained, starting from a bootstrap stage and gradually enhancing the representation power of both as the training proceeds.

In detail, we co-train the models by mining positive pairs from the other view of data. The RGB representation $f_1(\cdot)$ is updated with a Multi-Instance InfoNCE [Miech et al. 2020] loss (that covers our case of one or more actual positives within the positive set $\mathcal{P}_{1i}$ defined below):

$$\mathcal{L}_1 = -\mathbb{E}\left[\log \frac{\sum_{p\in\mathcal{P}_{1i}} \exp(z_{1i}\cdot z_p/\tau)}{\sum_{p\in\mathcal{P}_{1i}} \exp(z_{1i}\cdot z_p/\tau) + \sum_{n\in\mathcal{N}_{1i}} \exp(z_{1i}\cdot z_n/\tau)}\right] \tag{4.2}$$

where the numerator is defined as a sum of 'similarity' between sample x_{1i} (in the RGB view) and a positive set, constructed by the video clips that are most similar

to x_{2i} (most similar video clips in the optical flow view):

$$\mathcal{P}_{1i} = \{\psi(x_{1i}; a), x_{1k} | k \in \text{topK}(z_{2i} \cdot z_{2j}), \forall j \in [1, N], a \sim A\} \tag{4.3}$$

$z_{2i} \cdot z_{2j}$ refers to the similarity between i-th and j-th video in the optical flow view, and the topK$(\cdot)$ operator selects the top$K$ items over all available N samples and returns their indexes. The K is a hyper parameter representing the strictness of positive mining. The negative set $\mathcal{N}_{1i}$ for sample x_i is the complement of the positive set, $\mathcal{N}_{1i} = \overline{\mathcal{P}_{1i}}$. In other words, the positive set consists of the top K nearest neighbours in the optical flow feature space plus the video clip's own augmentations, and the negative set contains all other video clips, and their augmentations.

Similarly, to update the optical flow representation, $f_2(\cdot)$, we can optimize:

$$\mathcal{L}_2 = -\mathbb{E}\left[\log \frac{\sum_{p \in \mathcal{P}_{2i}} \exp(z_{2i} \cdot z_p / \tau)}{\sum_{p \in \mathcal{P}_{2i}} \exp(z_{2i} \cdot z_p / \tau) + \sum_{n \in \mathcal{N}_{2i}} \exp(z_{2i} \cdot z_n / \tau)}\right] \tag{4.4}$$

It is an identical objective function to (4.2) except that the positive set is now constructed from similarity ranking in the RGB view:

$$\mathcal{P}_{2i} = \{\psi(_{2i}; a), x_{2k} | k \in \text{topK}(z_{1i} \cdot z_{1j}), \forall j \in [1, N], a \sim A\} \tag{4.5}$$

The CoCLR algorithm proceeds in two stages: initialization and alternation. *Initialization.* To start with, the two models with different views are trained independently with InfoNCE, i.e. the RGB and Flow networks are trained by optimizing $\mathcal{L}_{\text{InfoNCE}}$.

Alternation. Once trained with $\mathcal{L}_{\text{InfoNCE}}$, both the RGB and Flow networks have gained far stronger representations than randomly initialized networks. The co-training process then proceeds as described in Eq. 4.2 and Eq. 4.4, e.g. to optimize $\mathcal{L}_1$, we mine hard positive pairs with a Flow network; to optimize $\mathcal{L}_2$, the hard positive pairs are mined with a RGB network. These two optimizations are alternated: each time first mining hard positives from the other network, and then minimizing the loss for the network independently. As the joint optimization proceeds, and the representations become stronger, different (and harder) positives are retrieved.

The key hyper-parameters that define the alternation process are: the value

of K used to retrieve the K semantically related video clip, and the number of iterations (or epochs) to minimize each loss function, i.e. the granularity of the alternation. These choices are explored in the ablations of Section 4.4.4, where it will be seen that a choice of $K = 5$ is optimal and more cycle alternations are beneficial; where each cycle refers to a complete optimization of $\mathcal{L}_1$ and $\mathcal{L}_2$; meaning, the alternation only happens after the RGB or Flow network has converged.

Discussion. *First*, when compared with our previous work that used InfoNCE for video self-supervision, DPC and MemDPC [Han et al. 2019; Han et al. 2020a], the proposed CoCLR incorporates learning from potentially harder positives, e.g. instances from the same class, rather than from only different augmentations of the same instance; *Second*, CoCLR differs from the *oracle* proposals of UberNCE since both the CoCLR positive and negative sets may still contain 'label' noise, i.e. class-wise false positives and false negatives. However, in practice, the Multi-Instance InfoNCE used in CoCLR is fairly robust to noise. *Third*, CoCLR is fundamentally different to two concurrent approaches, CMC [Yonglong Tian et al. 2020a] and CVRL [Qian et al. 2020], that use only instance-level training, i.e. positive pairs are constructed from the same data sample. Specifically, CMC extends positives to include different views, RGB and flow, of the same video clip, but does not introduce positives between clips; CVRL uses InfoNCE contrastive learning with video clips as the instances. We present experimental results for both InfoNCE and CMC in Table 4.1.

4.4 Experiments

In this section, we first describe the datasets (Section 4.4.1) and implementation details (Section 4.4.2) for CoCLR training. In Section 4.4.3, we describe the downstream tasks for evaluating the representation obtained from self-supervised learning. All proof-of-concept and ablation studies are conducted on UCF101 (Section 4.4.4), with larger scale training on Kinetics-400 (Section 4.4.5) to compare with other state-of-the-art approaches.

4.4.1 Datasets

We use two video action recognition datasets for self-supervised CoCLR training: **UCF101** [Soomro et al. 2012], containing 13k videos spanning 101 human actions (we only use the videos from the training set); and Kinetics-400 (**K400**) [Kay et al. 2017] with 240k video clips only from its training set. For downstream evaluation tasks, we benchmark on the UCF101 split1, K400 validation set, as well as on the split1 of **HMDB51** [Hildegard Kuehne et al. 2011], which contains 7k videos spanning 51 human actions.

4.4.2 Implementation Details for CoCLR

We choose the S3D [S. Xie et al. 2018] architecture as the feature extractor for all experiments. During CoCLR training, we attach a non-linear projection head, and remove it for downstream task evaluations, as done in SimCLR [T. Chen et al. 2020]. We use a 32-frame RGB (or flow) clip as input, at 30 fps, this roughly covers 1 second. The video clip has a spatial resolution of 128×128 pixels. For data augmentation, we apply random crops, horizontal flips, Gaussian blur and color jittering, all are clip-wise consistent. We also apply random temporal cropping to utilize the natural variation of the temporal dimension, i.e. the input video clips are cropped at random time stamps from the source video. The optical flow is computed with the *un-supervised* TV-L1 algorithm [Zach et al. 2007], and the same pre-processing procedure is used as in [João Carreira and Zisserman 2017]. Specifically, two-channel motion fields are stacked with a third zero-valued channel, large motions exceeding 20 pixels are truncated, and the values are finally projected from $[-20, 20]$ to $[0, 255]$ then compressed as jpeg.

At the *initialization* stage, we train both RGB and Flow networks with InfoNCE for 300 epochs, where an epoch means to have sampled one clip from each video in the training set, i.e. the total number of seen instances is equivalent to the number of videos in the training set. We adopt a momentum-updated history queue to cache a large number of features as in MoCo [K. He et al. 2020; X. Chen et al. 2020]. At the *alternation* stage, on UCF101 the model is trained for two cycles, where each cycle includes 200 epochs, i.e. RGB and Flow networks are each trained for 100 epochs with hard positive mining from the other; on K400 the model is only trained for one cycle for 100 epochs, that is 50 epochs each for RGB

and Flow networks, however, we expect more training cycles to be beneficial. For optimization, we use Adam with 10^{-3} learning rate and 10^{-5} weight decay. The learning rate is decayed down by 1/10 twice when the validation loss plateaus. Each experiment is trained on 4 GPUs, with a batch size of 32 samples per GPU.

4.4.3 Downstream tasks for representation evaluation

Action classification. In this protocol, we evaluate on two settings: (1) **linear probe**: the entire feature encoder is frozen, and only a single linear layer is trained with cross-entropy loss, (2) **finetune**: the entire feature encoder and a linear layer are finetuned end-to-end with cross-entropy loss, i.e. the representation from CoCLR training provides an initialization for the network.

At the training stage, we apply the same data augmentation as in the pre-training stage mentioned in Section 4.4.2, except for the Gaussian blur. At the inference stage, we follow the same procedure as our previous work [Han et al. 2019; Han et al. 2020a]: for each video we spatially apply ten-crops (center crop plus four corners, with horizontal flipping) and temporally take clips with moving windows (half temporal overlap), and then average the predicted probabilities.

Action retrieval. In this protocol, the extracted feature is directly used for nearest-neighbour (NN) retrieval and no further training is allowed. We follow the common practice [D. Xu et al. 2019; D. Luo et al. 2020], and use testing set video clips to query the k-NNs from the training set. We report Recall at k (R@k), meaning, if the top k nearest neighbours contains one video of the same class, a correct retrieval is counted.

4.4.4 Model comparisons on UCF101

This section demonstrates the evolution from InfoNCE to UberNCE and to CoCLR, and we monitor the top1 accuracy of action classification and retrieval performance. In this section, the *same* dataset, UCF101 split # 1 is used for self-supervised training and downstream evaluations, and we mainly focus on the linear probe & retrieval as the primary measures of representation quality, since their evaluation is fast. For all self-supervised pretraining, we keep the settings identical, e.g. training epochs, and only vary the process for mining positive pairs.

Pretrain Stage			Classification Top1		Retrieval
Method	Input	Labels	Linear probe	Finetune	R@1
InfoNCE	RGB	✗	46.8	78.4	33.1
InfoNCE	Flow	✗	66.8	83.1	45.2
UberNCE	RGB	✓	78.0	80.0	71.6
Cross-Ent.	RGB	✓	-	77.0⋆	73.5
CMC§	RGB	✓	55.0	-	-
$\text{CoCLR}_{\text{K=5}}$	RGB	✗	70.2	81.4	51.8
$\text{CoCLR}_{\text{K=5}}$	Flow	✗	68.7	83.5	48.4
$\text{CoCLR}_{\text{K=5}}$†	R+F	✗	**72.1**	**87.3**	**55.6**
$\text{CoCLR}_{\text{K=1}}$	RGB	✗	60.5	79.5	48.5
$\text{CoCLR}_{\text{K=50}}$	RGB	✗	68.3	81.0	49.8
$\text{CoCLR}_{\text{K=5, sim}}$	RGB	✗	65.2	80.8	48.0

Table 4.1: Representations from InfoNCE, UberNCE and CoCLR are evaluated on downstream action classification and retrieval. Left refers to the setting for pre-training. CMC§ [Yonglong Tian et al. 2020a] is our implementation for a fair comparison to CoCLR, i.e. S3D architecture, trained with 500 epochs. † refers to the results from two-stream networks (RGB + Flow). ⋆Cross-Ent. is end-to-end training with Softmax Cross-Entropy.

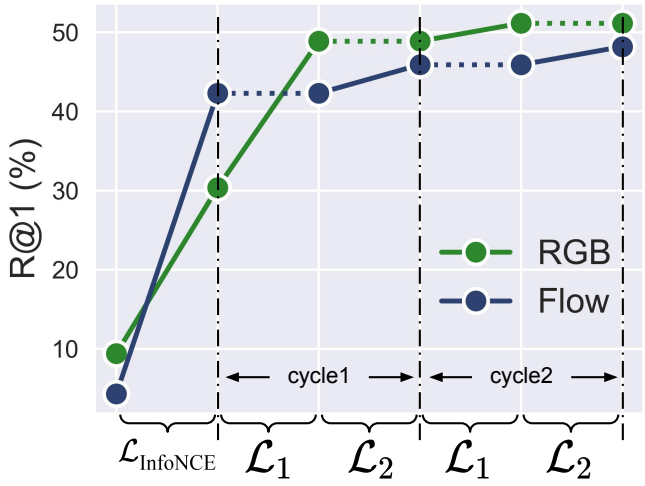


Figure 4.2: Training progress of CoCLR on UCF101 dataset for RGB and optical flow input, i.e. R@1 of training set video retrieval for both RGB and Flow, the dotted line means that the representation is fixed at certain training stage.

Linear probe & retrieval. The discussion here will focus on the RGB network, as this network is easy to use (no flow computation required) and offers fast inference speed, but training was done with both RGB and Flow for CoCLR and CMC. As shown in Table 4.1, three phenomena can be observed: *First*, UberNCE, the *supervised* contrastive method outperforms the InfoNCE baseline with a significant gap on the linear probe (78.0 vs 46.8) and retrieval (71.6 vs 33.1), which reveals the suboptimality of the *instance-based* self-supervised learning. *Second*, the co-training scheme (CoCLR) shows its effectiveness by substantially improving

over the InfoNCE and CMC baselines from 46.8 and 55.0 to 70.2, approaching the results of UberNCE (78.0). *Third*, combining the logits from both the RGB and Flow networks (denoted as CoCLR†) brings further benefits. We conjecture that a more modern RGB network, such as SlowFast [Feichtenhofer et al. 2019b], that is able to naturally capture more of the motion information, would close the gap even further.

End-to-end finetune. In this protocol, all models (RGB networks) are performing similarly well, and the gaps between different training schemes are marginal (77.0 – 81.4). This is expected, as the same dataset, data view and architecture have been used for self-supervised learning, finetuning or training from scratch. In this paper, we are more interested in the scenario, where pretraining is conducted on a large-scale dataset, e.g. Kinetics, and feature transferability is therefore evaluated on linear probing and finetuning on another small dataset, as demonstrated in Section 4.4.5.

For a better understanding of the effectiveness of co-training on mining hard positive samples, in Figure 4.2, we monitor the *alternation* process by measuring R@1. Note that, the label information is only used to plot this curve, but *not* used during self-supervised training. The x-axis shows the training stage, initialized from the InfoNCE representation, followed by alternatively training $\mathcal{L}_1$ and $\mathcal{L}_2$ for two cycles, as explained in Section 4.3.2. The dotted line indicates that a certain network is fixed, and the solid line indicates that the representation is being optimized. As training progresses, the representation quality of both the RGB and Flow models improve with more co-training cycles, shown by the increasing R@1 performance, which indicates that the video clips with same class have indeed been pulled together in the embedding space.

Ablations. We also experimented with other choices for the CoCLR hyper-parameters, and report the results at the bottom Table 4.1. In terms of number of the samples mined in Eq. 4.3 and Eq. 4.5, $K = 5$ is the optimal setting, i.e. the the Top5 most similar samples are used to train the target representation. Other values, $K = 1$ and $K = 50$ are slightly worse. In terms of *alternation* granularity, we compare with the extreme case that the two representations are optimized simultaneously ($\text{CoCLR}_{\text{K=5; sim}}$), again, this performs slightly worse than training one network with the other fixed to 'act as' an oracle. We conjecture that the inferior performance of simultaneous optimization is because the weights of the

network are updated too fast, similar phenomena have also been observed in other works [Sutton et al. 2000; K. He et al. 2020], we leave further investigation of this to future work.

4.4.5 Comparison with the state-of-the-art

In this section, we compare CoCLR with previous self-supervised approaches on action classification. Specifically, we provide results of CoCLR under two settings, namely, trained on UCF101 with $K = 5$ for 2 cycles; and on K400 with $K = 5$ for 1 cycle only. Note that there has been a rich literature on video self-supervised learning, and in Table 4.2 we only list some of the recent approaches evaluated on the same benchmark, and try to compare with them as fairly as we can, in terms of architecture, training data, resolution (although there remain variations).

In the following we compare with the methods that are trained with: (i) visual information only on the same training set; (ii) visual information only on larger training sets; (iii) multimodal information.

Visual-only information with same training set (finetune). When comparing the models that are only trained (both self-supervised and downstream finetune) on **UCF101**, e.g. OPN and VCOP, the proposed CoCLR (RGB network) obtains Top1 accuracy of 81.4 on UCF101 and 52.1 on HMDB, significantly outperforming all previous approaches. Moving onto **K400**, recent approaches include 3D-RotNet, ST-Puzzle, DPC, MemDPC, XDC, GDT, and SpeedNet. Again, CoCLR (RGB network) surpasses the other self-supervised methods, achieving 87.9 on UCF101 and 54.6 on HMDB, and the two-stream CoCLR† brings further benefits (90.6 on UCF101 and 62.9 on HMDB). We note that CoCLR is slightly underperforming CVRL [Qian et al. 2020], which we conjecture is due to the fact that CVRL has been trained with a deeper architecture (23 vs. 49 layers) with more parameters (7.9M vs. 33.1M), larger resolution (128 vs. 224), stronger color jittering, and far more epochs (400 vs. 800 epochs). This also indicates that potentially there remains room for further improving CoCLR, starting from a better initialization trained with InfoNCE.

Visual-only information with larger training set (finetune). Although only visual information is used, some approaches exploit a larger training set, e.g. CBT and DynamoNet. CoCLR† still outperforms all these approaches, showing its remarkable training efficiency, in the sense that it can learn better

representation with far less data.

Multi-modal information (finetune). These are the methods that exploit the correspondence of visual information with text or audio. The methods usually train on much larger-scale datasets, for instance, AVTS trained on AudioSet (8x larger than K400), and XDC trained on IG65M (273x larger than K400), for audio-visual correspondence; MIL-NCE is trained on narrated instructional videos (195x larger than K400) for visual-text correspondence; and ELO [Piergiovanni et al. 2020] is trained with 7 different losses on 2 million videos (104x larger than K400). Despite these considerably larger datasets, and information from other modalities, our best visual-only CoCLR† (two-stream network) still compares favorably with them. Note that, our CoCLR approach is also not limited to visual-only self-supervised learning, and is perfectly applicable for mining hard positives from audio or text.

Linear probe. As shown in the upper part of Table 4.2, CoCLR outperforms MemDPC and CBT significantly, with the same or only a tiny proportion of data for self-supervised training, and compares favorably with MIL-NCE, XDC and ELO that are trained on orders of magnitude more training data.

Video retrieval. In addition to the action classification benchmarks, we also evaluate CoCLR on video retrieval, as explained in Section 4.4.3. The goal is to test if the query clip instance and its nearest neighbours belong to same semantic category. As shown in Table 4.3, in both benchmark datasets, the InfoNCE baseline models exceed all previous approaches by a significant margin. Our CoCLR models further exceed InfoNCE models by a large margin.

Qualitative results for video retrieval. Figure 4.3 visualizes a query video clip and its Top3 Nearest Neighbors from the UCF101 training set using the CoCLR embedding. As can be seen, the representation learnt by CoCLR has the ability to retrieve videos with the same semantic categories.

Method	Date	Dataset (duration)	Res.	Arch.	Depth	Modality	Frozen	UCF	HMDB
CBT [C. Sun et al. 2019]	2019	K600+ (273d)	112	S3D	23	V	✓	54.0	29.5
MemDPC [Han et al. 2020a]	2020	K400 (28d)	224	R-2D3D	33	V	✓	54.1	30.5
MIL-NCE [Miech et al. 2020]	2020	HTM (15y)	224	S3D	23	V+T	✓	82.7	53.1
MIL-NCE [Miech et al. 2020]	2020	HTM (15y)	224	I3D	22	V+T	✓	83.4	54.8
XDC [Alwassel et al. 2019]	2019	IG65M (21y)	224	R(2+1)D	26	V+A	✓	85.3	56.0
ELO [Piergiovanni et al. 2020]	2020	Youtube8M- (8y)	224	R(2+1)D	65	V+A	✓	–	64.5
CoCLR-RGB		UCF (1d)	128	S3D	23	V	✓	70.2	39.1
CoCLR-2Stream†		UCF (1d)	128	S3D	23	V	✓	72.1	40.2
CoCLR-RGB		K400 (28d)	128	S3D	23	V	✓	74.5	46.1
CoCLR-2Stream†		K400 (28d)	128	S3D	23	V	✓	77.8	52.4
OPN [Lee et al. 2017]	2017	UCF (1d)	227	VGG	14	V	✗	59.6	23.8
3D-RotNet [Jing and Yingli Tian 2018]	2018	K400 (28d)	112	R3D	17	V	✗	62.9	33.7
ST-Puzzle [Kim et al. 2019]	2019	K400 (28d)	224	R3D	17	V	✗	63.9	33.7
VCOP [D. Xu et al. 2019]	2019	UCF (1d)	112	R(2+1)D	26	V	✗	72.4	30.9
DPC [Han et al. 2019]	2019	K400 (28d)	128	R-2D3D	33	V	✗	75.7	35.7
CBT [C. Sun et al. 2019]	2019	K600+ (273d)	112	S3D	23	V	✗	79.5	44.6
DynamoNet [Diba et al. 2019]	2019	Youtube8M-1 (58d)	112	STCNet	133	V	✗	88.1	59.9
SpeedNet [Benaim et al. 2020]	2020	K400 (28d)	224	S3D-G	23	V	✗	81.1	48.8
MemDPC [Han et al. 2020a]	2020	K400 (28d)	224	R-2D3D	33	V	✗	86.1	54.5
CVRL [Qian et al. 2020]	2020	K400 (28d)	224	R3D	49	V	✗	92.1	65.4
AVTS [Korbar et al. 2018]	2018	K400 (28d)	224	I3D	22	V+A	✗	83.7	53.0
AVTS [Korbar et al. 2018]	2018	AudioSet (240d)	224	MC3	17	V+A	✗	89.0	61.6
XDC [Alwassel et al. 2019]	2019	K400 (28d)	224	R(2+1)D	26	V+A	✗	84.2	47.1
XDC [Alwassel et al. 2019]	2019	IG65M (21y)	224	R(2+1)D	26	V+A	✗	94.2	67.4
GDT [Patrick et al. 2020]	2020	K400 (28d)	112	R(2+1)D	26	V+A	✗	89.3	60.0
GDT [Patrick et al. 2020]	2020	G65M (21y)	112	R(2+1)D	26	V+A	✗	95.2	72.8
MIL-NCE [Miech et al. 2020]	2020	HTM (15y)	224	S3D	23	V+T	✗	91.3	61.0
ELO [Piergiovanni et al. 2020]	2020	Youtube8M-2 (13y)	224	R(2+1)D	65	V+A	✗	93.8	67.4
CoCLR-RGB		UCF (1d)	128	S3D	23	V	✗	81.4	52.1
CoCLR-2Stream†		UCF (1d)	128	S3D	23	V	✗	87.3	58.7
CoCLR-RGB		K400 (28d)	128	S3D	23	V	✗	87.9	54.6
CoCLR-2Stream†		K400 (28d)	128	S3D	23	V	✗	90.6	62.9
Supervised [S. Xie et al. 2018]		K400 (28d)	224	S3D	23	V	✗	96.8	75.9

Table 4.2: Comparison with state-of-the-art approaches. In the left columns, we show the pre-training setting, e.g. dataset, resolution, architectures with encoder depth, modality. In the right columns, the top-1 accuracy is reported on the downstream action classification task for different datasets, e.g. UCF, HMDB, K400. The dataset parenthesis shows the total video duration in time (**d** for day, **y** for year). 'Frozen ✗' means the network is end-to-end finetuned from the pretrained representation, shown in the top half of the table; 'Frozen ✓' means the pretrained representation is fixed and classified with a linear layer, shown in the bottom half. For input, 'V' refers to visual only (colored with blue), 'A' is audio, 'T' is text narration. CoCLR models with † refer to the two-stream networks, where the predictions from RGB and Flow networks are averaged.

Method	Date	Dataset	UCF				HMDB			
			R@1	R@5	R@10	R@20	R@1	R@5	R@10	R@20
Jigsaw	2016	UCF	19.7	28.5	33.5	40.0	-	-	-	-
OPN	2017	UCF	19.9	28.7	34.0	40.6	-	-	-	-
Buchler	2018	UCF	25.7	36.2	42.2	49.2	-	-	-	-
VCOP	2019	UCF	14.1	30.3	40.4	51.1	7.6	22.9	34.4	48.8
VCP	2020	UCF	18.6	33.6	42.5	53.5	7.6	24.4	36.3	53.6
MemDPC	2020	UCF	20.2	40.4	52.4	64.7	7.7	25.7	40.6	57.7
SpeedNet	2020	K400	13.0	28.1	37.5	49.5	-	-	-	-
InfoNCE-RGB		UCF	36.0	52.0	61.8	71.0	15.2	34.7	48.9	63.2
InfoNCE-Flow		UCF	45.5	67.5	75.4	**82.7**	21.4	46.3	**59.6**	**72.1**
CoCLR-RGB		UCF	53.3	69.4	76.6	82.0	23.2	43.2	53.5	65.5
CoCLR-Flow		UCF	51.9	68.5	75.0	80.8	23.9	**47.3**	58.3	69.3
CoCLR-2Stream†		UCF	**55.9**	**70.8**	**76.9**	82.5	**26.1**	45.8	57.9	69.7

Table 4.3: Comparison with others on Nearest-Neighbour video retrieval on UCF101 and HMDB51. Full reference list: Jigsaw [Noroozi and Favaro 2016], OPN [Lee et al. 2017], Buchler [Büchler et al. 2018], VCOP [D. Xu et al. 2019], VCP [D. Luo et al. 2020], MemDPC [Han et al. 2020a], SpeedNet [Benaim et al. 2020]. Testing set clips are used to retrieve training set videos and R@k is reported, where $k \in [1, 5, 10, 20]$. Note that all the models reported were only pretrained on UCF101 with self-supervised learning except SpeedNet. † For two-stream network, the feature similarity scores from RGB and Flow networks are averaged.

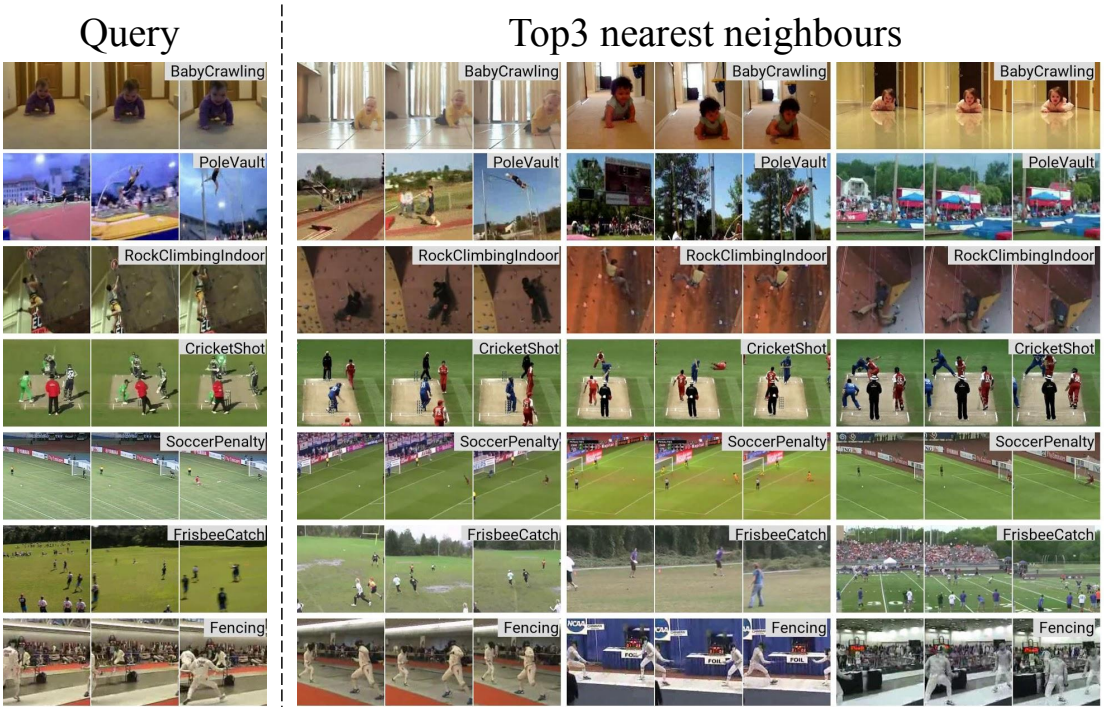


Figure 4.3: Nearest neighbour retrieval results with CoCLR representations. The left side is the query video from the UCF101 testing set, and the right side are the top 3 nearest neighbours from the UCF101 training set. CoCLR is trained only on UCF101. The action label for each video is shown in the upper right corner.

4.5 Conclusion

We have shown that a complementary view of video can be used to bridge the gap between RGB video clip instances of the same class, and that using this to generate positive training sets substantially improves the performance over InfoNCE instance training for video representations. Though we have not shown it in this paper, we conjecture that *explicit* mining from audio can provide a similar role to optical flow. For example, the sound of a guitar can link together video clips with very different visual appearances, even if the audio network is relatively untrained. This observation in part explains the success of audio-visual self-supervised learning, e.g. [Arandjelović and Zisserman 2017; Arandjelović and Zisserman 2018; Korbar et al. 2018; Alwassel et al. 2019; Patrick et al. 2020] where such links occur *implicitly*. Similarly and more obviously, text provides the bridge between instances in visual-text learning, e.g. from videos with narrations that describe their visual content [Miech et al. 2020]. We expect that the success of explicit positive mining in CoCLR will lead to applications to other data, e.g. images, other modalities and tasks where other views can be extracted to provide complementary information, and also to other learning methods, such as BYOL [Grill et al. 2020].

4.6 Broader Impact

Deep learning systems are data-hungry and are often criticized for their huge financial and environmental cost. Training a deep neural network end-to-end is especially expensive due to the large computational requirements. Our research on video representation learning has shown its effectiveness on various downstream tasks. As a positive effect of this, future research can benefit from our work by building systems with the pretrained representation to save the cost of re-training. However, on the negative side, research on self-supervised representation learning has consumed many computational resources and we hope more efforts are put on reducing the training cost in this research area. To facilitate future research, we release our code and pretrained representations.

Acknowledgement

We thank Triantafyllos Afouras, Andrew Brown, Christian Rupprecht and Chuhan Zhang for proofreading and helpful discussions. Funding for this research is provided by a Google-DeepMind Graduate Scholarship, and by the EPSRC Programme Grant Seebibyte EP/M013774/1.

Appendices

Appendices for this chapter can be found in the online version of the paper. [1]

Statement of authorship

A statement of authorship for this paper is provided in Appendix A.

[1] https://www.robots.ox.ac.uk/~vgg/research/CoCLR/

Part II

Efficient Video Representation Learning

Chapter 5

Prompting Visual-Language Models for Efficient Video Understanding

The paper has been accepted for publication at the European Conference on Computer Vision (ECCV), 2022.

Prompting Visual-Language Models for Efficient Video Understanding

Chen Ju[1], Tengda Han[2], Kunhao Zheng[1], Ya Zhang[1], Weidi Xie[2]

[1]Cooperative Medianet Innovation Center, Shanghai Jiao Tong University

[2]VGG, Department of Engineering Science, University of Oxford

{ju_chen, dyekuu, ya_zhang}@sjtu.edu.cn {htd, weidi}@robots.ox.ac.uk

https://ju-chen.github.io/efficient-prompt

Abstract

Visual-language pre-training has shown great success for learning joint visual-textual representations from large-scale web data, demonstrating remarkable ability for "zero-shot" generalisation. This paper presents a simple method to efficiently adapt one pre-trained visual-language model to novel tasks with minimal training, and here, we consider video understanding tasks. Specifically, we propose to optimise a few random vectors, termed as "continuous prompt vectors", that convert the novel tasks into the same format as the pre-training objectives. In addition, to bridge the gap between static images and videos, temporal information is encoded with lightweight Transformers stacking on top of frame-wise visual features. Experimentally, we conduct extensive ablation studies to analyse the critical components and necessities. On 9 public benchmarks of action recognition, action localisation, and text-video retrieval, across closed-set, few-shot, open-set scenarios, we achieve competitive or state-of-the-art performance to existing methods, despite training significantly fewer parameters.

5.1 Introduction

While research in computer vision has mainly focused on tackling particular tasks, the grand goal towards human-level perception has always been to learn general-purpose visual representation, that can solve various problems with *minimal tunings*. Towards such goal, recent work for training visual-language models has shown promising progress. For example, CLIP [Radford et al. 2021] and ALIGN [C. Jia et al. 2021] learn a joint representation for image and text with simple noise contrastive learning, greatly benefiting from the rich information in text descriptions, *e.g.* actions, objects, human-object interactions, and object-object relationships. As a result, these visual-language models have demonstrated remarkable "zero-shot" generalisation for various image classification tasks. Crucially, the data used to train these powerful visual-language models can simply be crawled from the Internet at scale, without any laborious manual annotation. It is therefore reasonable to believe, with the growing computation, larger datasets will be collected, and more powerful models will be trained in the near future.

Given the promise, one question naturally arises: *how can we best exploit the ability in these powerful visual-language models, and effectively adapt it to solve particular novel vision tasks that are of interest?* One possible solution would be to finetune the image encoder end-to-end on the considered downstream tasks, however, as each downstream task will need to finetune and save its own set of parameters, we end up developing hundreds of models for hundreds of individual tasks. Even more problematic, discarding the text encoder loses the model's ability for "zero-shot" generalisation, thus the resultant model can only work for a fixed set of pre-determined categories. Alternatively, as shown in CLIP [Radford et al. 2021], given properly designed "prompts", the model is able to work on a variety of downstream tasks, with the classifiers being dynamically generated by the text encoder, from category names or other free-form texts. The prompts here are handcrafted cloze templates to facilitate classifier generation, so that downstream visual tasks can be formulated in the same format as the pre-training objectives, effectively closing the gap between pre-training and downstream tasks. One remaining issue is, such handcrafted prompts require extensive expert knowledge and labor, limiting its use for efficient task adaptation.

In this paper, we continue in the vein of prompt-based learning, with the goal

of exploring an *efficient* way to adapt a visual-language model for novel tasks. We here consider a simple idea by prepending / appending a sequence of random vectors, which are termed as "continuous prompt vectors", to the textual input. These prompt vectors consist entirely of free parameters that do not correspond to any real concrete words, and the subsequent layers of the text encoder will attend these vectors, as if they were a sequence of "virtual tokens" to generate the classifier or embedding. Although the weights of text encoder are kept frozen, the gradients are back-propagated through it to optimise the trainable prompt vectors. Consequently, a single copy of the visual backbone is able to perform various video understanding tasks, with the minimal number of trainable parameters for each task, *i.e.*, only a few prompt vectors.

To summarise, building on a scalable, powerful image-based visual-language model, we first propose a simple idea for the *efficient* and *lightweight* model adaptation, through learning task-specific prompt vectors. And here, we consider video understanding as the downstream tasks, *e.g.* action recognition, action localisation, text-video retrieval; In particular, we formulate the problem of action recognition and retrieval under the same umbrella, that is, to maximise the similarity matching between visual and textual embeddings, with texts being action labels or fine-grained descriptions respectively; We extensively analyse several critical components of our method, *e.g.* the number of prompt vectors, usefulness of temporal modeling; Lastly, we evaluate the adaptation idea on 9 public video benchmarks, across closed-set, few-shot, and open-set scenarios. Despite training only a few free parameters, *i.e.* several prompt vectors and two Transformer layers, we are able to achieve competitive or state-of-the-art performance to existing methods. In few-shot and open-set scenarios, we significantly outperform existing methods, sometimes by over 10%.

5.2 Related Work

Joint Visual-Textual Learning. In the literature, [Y. Mori et al. 1999] has explored the connection between images and words using paired text documents, and [Weston et al. 2011; Frome et al. 2013] proposed to jointly learn image-text embeddings with the class name annotations. Recently, CLIP [Radford et al. 2021]

and ALIGN [C. Jia et al. 2021] have further scaled up the training with large-scale web data. Using simple noise contrastive learning, it is shown that powerful visual representation can be learnt from paired image-caption. In video domains, similar idea has also been explored for representation learning [Miech et al. 2020] and video retrieval [Miech et al. 2018; J. Lei et al. 2021].

Prompting refers to designing proper "instructions" that a pre-trained language model can understand, and generate desired outputs, using a few examples as demonstrations. For instance, given properly handcrafted prompt templates, GPT-3 [T. Brown et al. 2020] has shown strong generalisations for few-shot or zero-shot learning. However, the handcrafted templates require extensive expert knowledge, limiting the flexibility. Later work proposes to automate the prompt engineering by searching discrete prompts [Z. Jiang et al. 2020; Shin et al. 2020; Schick and Schütze 2021; T. Gao et al. 2021], and continuous prompts [X. L. Li and P. Liang 2021; Lester et al. 2021]. In this work, we consider to search continuous prompts for steering pre-trained visual-language models to tackle video understanding tasks.

Video Action Recognition. In the last decade, research on developing effective architectures has gone through rapid developments, from two-stream networks [Simonyan and Zisserman 2014; Limin Wang et al. 2016; Feichtenhofer et al. 2016] to more recent single stream RGB networks [Joao Carreira and Zisserman 2017; Tran et al. 2018; S. Xie et al. 2018; Feichtenhofer et al. 2019b; Feichtenhofer 2020; Bertasius et al. 2021]. With the help of abundant training data, *e.g.* Kinetics [João Carreira et al. 2019], recognition accuracy has been steadily improved. In addition, researchers have also explored the ideas of data-efficient learning, for instance, few-shot and open-set action recognition. Specifically, in the former line of research, only a few training samples are available from each action category, [L. Zhu and Y. Yang 2018; L. Zhu and Y. Yang 2020b] proposed compound memory networks to classify videos by matching and ranking; [Dwivedi et al. 2019] used GANs to synthesize training examples for novel categories; [Cao et al. 2020] proposed differentiable dynamic time warping to align videos of different lengths; [Perrett et al. 2021] exploited CrossTransformer, to find temporally-corresponding frame tuples between the query and given few-shot videos. While in open-set action recognition, it requires the model to generalise towards action categories that are unseen in the training set, one typical idea lies in learning a common representation space

that is shared by seen and unseen actions, such as attributes space [J. Liu et al. 2011; Gan et al. 2016a], semantic space [Yikang Li et al. 2016; Gan et al. 2016b], synthesizing features to unseen actions [Mishra et al. 2020], using objects to create common space for unseen actions [Mettes et al. 2021].

Video Action Localisation considers to detect and classify actions of interest from untrimmed long videos. Generally speaking, there are two popular paradigms: the two-stage paradigm [Shou et al. 2016; Huijuan Xu et al. 2017; Y. Zhao et al. 2017; Chao et al. 2018; T. Lin et al. 2018; T. Lin et al. 2019; J. Tan et al. 2021; Ju et al. 2021] first detects class-agnostic action proposals, which cover correct segments with high recall, then classifies and refines each of these proposals. In contrast, the one-stage paradigm [Yeung et al. 2016; T. Lin et al. 2017; Nawhal and G. Mori 2021] combines action detection and classification, and densely classifies each timestamp into action or background.

Concurrent Work. Several recent arXiv papers [K. Zhou et al. 2021; P. Gao et al. 2021; Renrui Zhang et al. 2021] also consider the prompt learning for efficient transfer from pre-trained visual-language models to downstream tasks, *e.g.* image classification. In video domains, CLIP4Clip [H. Luo et al. 2021] and ActionCLIP [M. Wang et al. 2021] propose to end-to-end finetune pre-trained CLIP on individual video tasks, *e.g.* retrieval and action recognition. In contrast, we favor the efficient adaptation from image to video, and aim to learn task-specific continuous prompts, to generate classifiers for various video understanding tasks under one same framework.

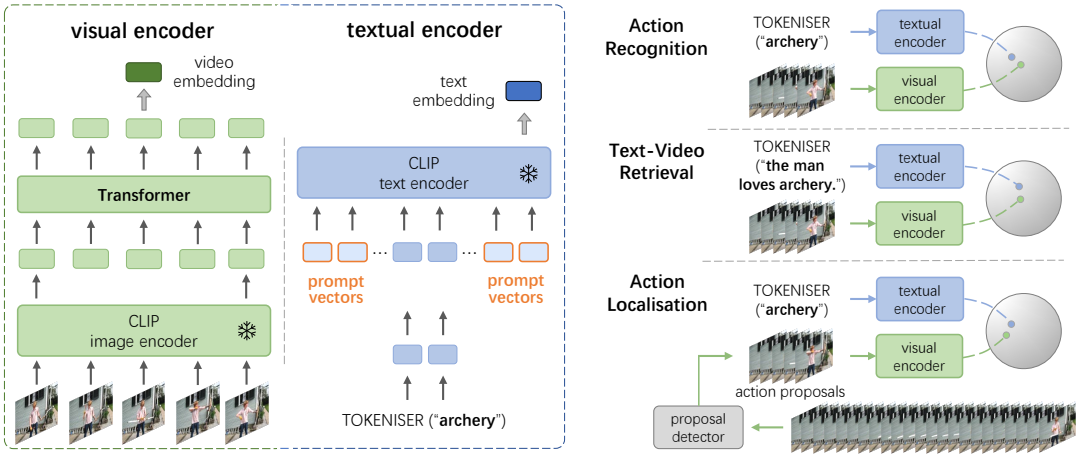


Figure 5.1: **Framework Overview**. We adopt a lightweight Transformer module on top of the CLIP image encoder for temporal modeling; prepend / append learnable continuous prompt vectors to the CLIP text encoder for generating action classifiers or text query embeddings. During training, both image and text encoders of CLIP are kept *frozen*. By optimising the task-specific prompt vectors and temporal Transformer module, we efficiently adapt CLIP to various video understanding tasks: action recognition, action localisation, and text-video retrieval, across closed-set, few-shot, and open-set scenarios.

5.3 Method

Our goal is to efficiently steer an image-based visual-language model to tackle novel downstream tasks, which we term as model adaptation. Here, we consider the downstream task to be video understanding, *i.e.* action recognition, localisation, and video retrieval. To be self-contained, in Section 5.3.1, we briefly review the pre-training and inference procedure of the original CLIP [Radford et al. 2021]; in Section 5.3.2, we describe the proposed idea for learning task-specific prompt vectors, and temporal modeling.

5.3.1 Visual-Language Model: CLIP

Pre-training. Given N (image, text) pairs in one sampling batch, the feature embeddings for image and text are computed with two encoders respectively, and dense cosine similarity matrix is calculated between all N possible (image, text) pairs. The training objective is to jointly optimise the image and text encoders, by maximizing the similarity between N correct pairs of (image, text) associations, while minimizing the similarity for $N \times (N-1)$ incorrect pairs via a symmetrical cross-entropy over the dense matrix, *i.e.* noise contrastive learning.

Note that, both image and text encoders contain a **tokeniser** for converting

image patches or language words to vectors. In particular, the input images are divided into patches and then flattened into vectors, also called "visual tokens"; while the input texts are converted into vectors ("textual tokens") by a trainable look-up table.

Inference. Once trained, CLIP can be deployed for image classification tasks on open vocabulary (zero-shot generalisation), with the visual classifiers being generated from the text encoder (Φ_{text}), which resembles the idea of hypernetwork [Ha et al. 2016]. For example, to classify an image as cat or dog, the classifiers (c_{cat} and c_{dog}) can be generated as:

$$c_{\text{cat}} = \Phi_{\text{text}}(\text{TOKENISER}(\text{“this is a photo of [cat]”}))$$
$$c_{\text{dog}} = \Phi_{\text{text}}(\text{TOKENISER}(\text{“this is a photo of [dog]”}))$$

and "this is a photo of [$\cdot$]" is a handcrafted prompt template, which has shown to be effective for image classification.

Discussion. Despite the tremendous success on zero-shot image classification, CLIP has also shown to be sensitive to the handcrafted prompt template, clearly posing limitations on its efficient adaptation for novel downstream tasks, where expert knowledge might be difficult to condense or unavailable. So here, we consider to automate such prompt design procedures, exploring efficient approaches to adapt the pre-trained visual-language models for the novel downstream video-related tasks, with minimal training.

5.3.2 Prompting CLIP for Video Understanding

In following sections, we start by describing the problem scenario and notations (Section 7.3.1); and introduce the idea for model adaptation through prompt learning (Section 5.3.2); lastly, we augment the CLIP image encoder with temporal modeling, disambiguating the events or actions that require temporal reasoning (Section 5.3.2).

Problem Scenario

Given a dataset that consists of training and validation sets, $\mathcal{D} = \{\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}\}$, *e.g.* $\mathcal{D}_{\text{train}} = \{(\mathcal{V}_1, y_1), \ldots, (\mathcal{V}_n, y_n)\}$. The video $\mathcal{V}_i \in \mathbb{R}^{T\times H\times W\times 3}$ can range from

seconds (recognition and retrieval), to minutes long (localisation). Respectively, y_i either refers to *one* of the $\mathcal{C}_{\text{train}}$ action labels in the text format for recognition, *e.g.* y_i = 'archery'; or *dense* action category labels of T timestamps for localisation, *e.g.* $y_i \in \mathbb{R}^{T \times \mathcal{C}_{\text{train}}}$; or fine-grained text descriptions for retrieval, *e.g.* y_i = 'fry the onion in a pan'.

In the closed-set scenario, the action categories for training and evaluation are the same, *i.e.* $\mathcal{C}_{\text{train}} = \mathcal{C}_{\text{val}}$; while in the open-set case, the action categories for training and evaluation are disjoint, *i.e.* $\mathcal{C}_{\text{train}} \cap \mathcal{C}_{\text{val}} = \emptyset$.

Model Adaptation by Learning Prompts

The goal here is to steer a pre-trained CLIP model to perform various video tasks with minimal training. In specific, we strive for efficient model adaptation by prepending / appending a sequence of continuous random vectors ("prompt vectors") with the textual tokens. While training, both the image and text encoders of CLIP are kept *frozen*, and the gradients will flow through the text encoder to only update the prompt vectors. Ultimately, these learned vectors end up constructing "virtual" prompt templates that can be understood by the text encoder, and to generate desired classifiers or query embeddings, as detailed below.

Action Recognition considers to classify the video clip or snippet into one of the action categories. In order to generate the classifier, we construct the "virtual" prompt template by feeding the tokenised action category name into the pre-trained text encoder (Φ_{text}), for instance:

$$c_{\text{archery}} = \Phi_{\text{text}}(a_1, \ldots, \text{TOKENISER}(\text{“}\underline{\text{archery}}\text{”}), \ldots, a_k)$$
$$c_{\text{bowling}} = \Phi_{\text{text}}(a_1, \ldots, \text{TOKENISER}(\text{“}\underline{\text{bowling}}\text{”}), \ldots, a_k)$$

where $a_i \in \mathbb{R}^D$ denotes i-th prompt vector, consisting of learnable parameters, and D is the vector dimension. c_{archery} refers to the generated classifier for the action of "archery". Note that, the prompt vectors $\{a_i\}$ are shared for all action categories, thus are only task-specific.

Action Localisation considers to localise and classify actions from untrimmed videos. Here, we leverage the two-stage paradigm [Chao et al. 2018; P. Zhao et al. 2020] to first detect potential class-agnostic action proposals (detailed in

Section 5.4.1), and followed by the action classification on these detected proposals.

Text-Video Retrieval considers to jointly learn the visual and textual embeddings that pair the video and its corresponding textual description. In contrast to action recognition, where a video snippet is coarsely labeled by an action category, the text description in video retrieval contains more fine-grained details, usually a sentence. We here similarly TOKENISE the entire sentence, and feed the tokenised results to the text encoder with learnable prompt vectors, to generate the *query embedding* for each sentence.

Summary. In general, learning prompts for model adaptation offers the following benefits: *First*, since both classification and retrieval can be tackled with one framework, with classifiers or query embeddings generated from text, either category names or free-form descriptions, all tasks can use *one* shared backbone, yet achieve competitive performance (Section 5.4); *Second*, adapting to novel tasks only requires to optimise a few prompt vectors, facilitating the few-shot problem (Section 5.4.2); *Third*, it enables to make better use of the abundant training data, and further generalise beyond the closed-set categories (Section 5.4.2).

Temporal Modeling

As for pre-training, CLIP has thoroughly relied on the (image, text) pairs, which poses clear pros on cons. On the one hand, the (image, text) data used for training can be easily crawled from the web, which enables to learn much richer contents under a given compute constraint; however, on the other hand, it ignores the temporal component of the visual scene, and struggles to recognise the dynamic events, *e.g.* push or pull, open or close. In this section, we bridge this image-to-video gap by adding a simple and lightweight temporal modeling module.

Specifically, we upgrade the CLIP image encoder Φ_{image} into a video one Φ_{video}, by attaching Transformers on top of frame-wise features from the frozen image encoder:

$$v_i = \Phi_{\text{video}}(\mathcal{V}_i) = \Phi_{\text{TEMP}}(\{\Phi_{\text{image}}(I_{i1}), \ldots, \Phi_{\text{image}}(I_{iT})\})$$

where Φ_{TEMP} refers to the temporal modeling module, which is a multi-layer Trans-

former Encoder, consisting of Multi-head Self-attention, Layer Norm, and MLPs. To indicate the temporal order, we also add learnable temporal positional encoding onto the image features. The $v_i \in \mathbb{R}^{T \times D}$ denotes dense feature embeddings of the T frames.

Training Loss

Given a batch of (video, text) pairs, the visual stream ends up with dense frame-wise feature embeddings (v_i); while for the textual stream, depending on the considered downstream tasks, it ends up with a set of action classifiers ($c_i \in \mathcal{C}_{\text{action}}$) or textual query embeddings ($c_i \in \mathcal{C}_{\text{query}}$).

For action recognition and text-video retrieval, we further compute the video-snippet-level feature by taking the mean pooling of the dense features:

$$\overline{v}_i = \Phi_{\text{POOL}}(v_i) \in \mathbb{R}^{1 \times D} \tag{5.1}$$

For action localisation, we take mean-pool of the dense features within each detected action proposal, to obtain the proposal-level feature (also denoted as $\overline{v}_i$ for simplicity).

During training, we jointly optimise the textual prompt vectors and temporal modeling module, such that the video (proposal) features and its paired classifier or textual query embedding emit the highest similarity score among others. This is achieved with a simple NCE loss:

$$\mathcal{L} = -\sum_i \Big(\log \frac{\exp(\overline{v}_i \cdot c_i / \tau)}{\sum_j \exp(\overline{v}_i \cdot c_j / \tau)} \Big) \tag{5.2}$$

Note that both $\overline{v}_i$ and c_j have been L2-normalised here, and τ refers to the temperature hyper-parameter for scaling. In this way, we effectively close the training objective gap between the CLIP pre-training and various video tasks.

5.4 Experiments

In this section, we conduct experiments on three different video understanding tasks, *i.e.* action recognition, action localisation, text-video retrieval, across 9 different datasets. In Section 5.4.2, we conduct ablation studies on action recognition,

to validate the usefulness of proposed components, *i.e.*, prompt learning and temporal modeling. In Section 5.4.3 and Section 5.4.4, we further benchmark on two other popular tasks: action localisation and text-video retrieval.

5.4.1 Implementation Details

In this paper, the image and text encoders are adopted from pre-trained CLIP (ViT-B/16+Transformer). For model adaptation, both encoders are kept frozen, the only trainable parts are the textual prompt vectors and temporal modeling module. All video frames are pre-processed to 224×224 spatial resolution, and the maximum number of textual tokens is 77 (follow the original CLIP design).

For action recognition, all videos are decoded to 30 fps, and 16 frames are sampled per video with a random frame gap (gap $\in \{1, 2, 3, 4, 5, 6, 10, 15\}$) for training. The temporal positional encodings consist of each frame's index and the frame sampling gap (that is, video playing speed). The model is optimised using AdamW [Ilya Loshchilov 2019] with a learning rate of 10^{-4}, and a batch size of 64 videos.

For action localisation, we follow a two-stage paradigm: class-agnostic action proposal detection and proposal classification. To obtain high-quality action proposals, we first divide the entire video into equal-frame snippets; then use the CLIP image encoder with one Transformer layer to extract frame-wise embeddings for each snippet; and finally feed these embeddings to the off-the-shelf proposal detectors [C. Lin et al. 2021; L. Yang et al. 2020]. These detectors construct the feature pyramid, and make predictions in parallel, to determine actionness, centerness, and boundaries. Please refer to [C. Lin et al. 2021; L. Yang et al. 2020] for detailed detector architectures and optimisations. Note that, our method is flexible to the choice of proposal detectors, and we do not innovate on such candidate proposal procedures. To generate proposal classifiers, we adopt the same implementation details as for action recognition.

For video retrieval, we also take the 16-frame input with a random frame gap (gap $\in \{10, 15, 30\}$). Note that, here we use significantly larger frame gaps than action recognition, as the retrieval task tends to require information from long-term visual dependencies. For more details, we refer the readers to the appendices.

5.4.2 Action Recognition

Datasets & Metrics. In this section, we conduct experiments on four popular benchmarks. **HMDB-51** [Hildegard Kuehne et al. 2011] contains 7k videos of 51 actions. Its standard split is to train on 3570 videos and evaluate on another 1530 videos. **UCF-101** [Soomro et al. 2012] contains 13k videos spanning 101 human actions. The standard split is to train on 9537 videos and evaluate on the left 3783 videos. **Kinetics-400** [Kay et al. 2017] (K-400) covers around 230k 10-second clips sourced from YouTube. Each clip describes one action category, and can be of different resolutions and frame rates. **Kinetics-700** [João Carreira et al. 2019] (K-700) is simply an extension of K-400, with around 650k video clips sourced from YouTube. To evaluate the recognition performance, we report the standard TOP1 and TOP5 accuracy, and the average of these two metrics.

Model	Prompt	Temporal	K-400			K-700		
			TOP1	TOP5	AVG	TOP1	TOP5	AVG
Baseline-I [Radford et al. 2021]	hand-craft	✗	–	–	–	–	–	52.4
Baseline-II [Radford et al. 2021]	✗	✗	–	–	–	–	–	66.1
A0	2+X+2	✗	65.4	88.7	77.1	56.3	81.9	69.1
A1	4+X+4	✗	66.1	89.0	77.6	56.6	82.4	69.5
A2	8+X+8	✗	67.9	90.0	79.0	57.4	83.0	70.2
A3	16+X+16	✗	68.8	90.1	79.5	57.8	83.1	70.5
A4	16+X+16	1-TFM	75.8	92.9	84.4	64.2	87.3	75.8
A5	16+X+16	2-TFM	76.6	93.3	85.0	64.7	88.5	76.6
A6	16+X+16	3-TFM	76.9	93.5	85.2	64.8	88.4	76.6
A7	16+X+16	4-TFM	76.8	93.5	85.2	64.9	87.9	76.4

Table 5.1: **Ablation study for closed-set action recognition**. Baseline-I is the "zero-shot" CLIP inference with handcrafted templates. Baseline-II is the standard linear probe on the pre-trained CLIP image encoder. TFM denotes the number of Transformer layers for temporal modeling.

Method	HMDB-51		UCF-101		K-400		K-700	
	TOP1	TOP5	TOP1	TOP5	TOP1	TOP5	TOP1	TOP5
I3D [Joao Carreira and Zisserman 2017]	74.3	–	95.1	–	71.6	90.0	58.7	81.7
S3D-G [S. Xie et al. 2018]	75.9	–	96.8	–	74.7	93.4	–	–
R(2+1)D [Tran et al. 2018]	74.5	–	96.8	–	72.0	90.0	–	–
TSM [J. Lin et al. 2019]	–	–	–	–	74.7	–	–	–
R3D-50 [Hara et al. 2018]	66.0	–	92.0	–	–	–	54.7	–
NL-I3D [X. Wang et al. 2018]	66.0	–	–	–	76.5	92.6	–	–
SlowFast [Feichtenhofer et al. 2019b]	–	–	–	–	77.0	92.6	–	–
X3D-XXL [Feichtenhofer 2020]	–	–	–	–	80.4	94.6	–	–
TimeSformer-L [Bertasius et al. 2021]	–	–	–	–	80.7	94.7	–	–
Ours (A5)	66.4	92.1	93.6	99.0	76.6	93.3	64.7	88.5

Table 5.2: **Comparison to state-of-the-art approaches on closed-set action recognition**. By training far fewer parameters, our model achieves comparable performance to the existing approaches on all datasets.

Method	K-shot	N-way	Prompt	Temporal	UCF-101	HMDB-51	K-400
TARN [Bishay et al. 2019]	5	5	–	–	–	–	78.5
ARN [H. Zhang et al. 2020]	5	5	–	–	83.1	60.6	82.4
OTAM [Cao et al. 2020]	5	5	–	–	–	–	85.8
TRX [Perrett et al. 2021]	5	5	–	–	96.1	75.6	85.9
Baseline-I [Radford et al. 2021]	–	5	hand-craft	✗	91.9	68.9	95.1
Ours	5	5	✓	✗	**98.3**	**85.3**	**96.4**
	5	5	✓	✓	97.8	84.9	96.0
Baseline-I [Radford et al. 2021]	–	$\mathcal{C}_{\text{ALL}}$	hand-craft	✗	64.7	40.1	54.2
Ours	5	$\mathcal{C}_{\text{ALL}}$	✓	✗	77.6	56.0	57.1
	5	$\mathcal{C}_{\text{ALL}}$	✓	✓	**79.5**	**56.6**	**58.5**

Table 5.3: **Comparison to state-of-the-art approaches on few-shot action recognition**. Here, $\mathcal{C}_{\text{ALL}}$ refers to the case where the model is tested on all categories of the corresponding dataset, rather than only 5-way classification, *e.g.* 101 categories for UCF, 400 categories for K-400. Baseline-I denotes the "zero-shot" CLIP inference with handcrafted templates.

Closed-set Action Recognition

Closed-set video action recognition refers to the commonly adopted setting, where the model is trained and evaluated on videos from the same action categories, *i.e.*, $\mathcal{C}_{\text{train}} = \mathcal{C}_{\text{val}}$. For comprehensive comparison, we here train and evaluate on the standard splits of four popular benchmarks, namely HMDB-51, UCF-101, K-400, and K-700.

Ablation Studies are conducted based on the two largest benchmark datasets, namely, K-400 and K-700. Table 5.1 presents the results for the prompt learning and temporal modeling module. And here, the prompt follows the format of $[a_1, .., a_k, \text{X}, a_{k+1}, .., a_{2k}]$. Note that, although we prepend and append equal number of prompt vectors, the optimisation can perfectly learn to ignore any of these vectors, thus we do not ablate the other prompt formats.

As baselines, we compare with the official results reported in the original CLIP [Radford et al. 2021]. Specifically, Baseline-I refers to the "zero-shot" inference with handcrafted prompt templates ("a photo of [·]."), and Baseline-II denotes the standard practice for training linear classifiers on top of the pre-trained image encoder on the downstream datasets.

Generally speaking, training more text prompt vectors brings consistent improvements on both TOP1 and TOP5 accuracy; In addition, adding temporal modeling also brings immediate benefits, with average gains of 4.9% and 5.3% on K-400 and K-700. However, it gives diminishing returns as more Transformer layers are added. Overall, all results suggest that, both the prompt learning and temporal modeling are essential. While comparing with Baseline-I, the A3 model demonstrates a performance boost of 18.1%, clearly showing the benefits of learn-

ing prompt vectors over handcrafted ones. Moreover, even with fewer trainable parameters, the A3 model also surpasses Baseline-II, with 4.4% gains, showing the superiority of prompting adaptation.

For all following action recognition experiments, we inherit the best practice from the ablation study, *i.e.* prepend / append 16 prompt vectors to category names, and only use two Transformer layers for temporal modeling, for its best trade-off on performance and computational cost.

Comparison to SOTA. Table 5.2 compares our model with the existing state-of-the-art approaches on the popular action recognition benchmarks. Overall, on all datasets, our model performs comparably with the competitors, although we only need to train *far fewer* parameters, *i.e.* two Transformer layers and several prompt vectors, advocating efficient model adaptation.

Few-shot Action Recognition

Few-shot action recognition aims to classify actions in the video with only a few samples, in this section, we consider to benchmark on two different settings. The first one follows the previous literature [H. Zhang et al. 2020; Perrett et al. 2021; Cao et al. 2020], and evaluates on the standard K-shot, N-way classification; while in the second part, we consider a more challenging setting that classifies all categories with K-shot training samples. For more details on dataset splits, please refer to the appendices. As baselines, in both settings, we use the "zero-shot" CLIP inference with handcrafted templates.

Model	Prompt	Temporal	Open-Set TOP1	Open-Set TOP5	Open-Set AVG
Baseline-I [Radford et al. 2021]	hand-craft	✗	52.4	77.3	64.9
B0	4+X+4	✗	57.4	83.3	70.4
B1	8+X+8	✗	57.7	82.6	70.2
B2	16+X+16	✗	**58.4**	82.6	70.5
B3	32+X+32	✗	57.5	**84.6**	**71.1**
B4	16+X+16	1-TFM	47.9	76.8	62.4
B5	16+X+16	2-TFM	45.5	75.4	60.5
B6	16+X+16	3-TFM	45.6	75.2	60.4

Table 5.4: **Ablation study for open-set action recognition on K-700**. Baseline-I refers to the results from CLIP open-set evaluation. The model is trained on 400 categories and tested on the other 300 disjoint categories.

5-Shot-5-Way Setting. For fair comparison, this setting adopts the publicly accessible few-shot splits. Specifically, for HMDB-51 and UCF-101, we follow [H. Zhang et al. 2020] to collect 10 and 21 testing action categories respectively; while

Method	Mode	THUMOS14 (mAP@IoU)						ActivityNet1.3 (mAP@IoU)			
		0.3	0.4	0.5	0.6	0.7	AVG	0.5	0.75	0.95	AVG
CDC [Shou et al. 2017]	RGB+Flow	40.1	29.4	23.3	13.1	7.9	22.8	45.3	26.0	0.2	23.8
TALNET [Chao et al. 2018]	RGB+Flow	53.2	48.5	42.8	33.8	20.8	39.8	38.2	18.3	1.3	20.2
BSN [T. Lin et al. 2018]	RGB+Flow	53.5	45.0	36.9	28.4	20.0	36.8	46.5	30.0	8.0	30.0
DBS [Z. Gao et al. 2019]	RGB+Flow	50.6	43.1	34.3	24.4	14.7	33.4	–	–	–	–
BUTAL [P. Zhao et al. 2020]	RGB+Flow	53.9	50.7	45.4	38.0	28.5	43.3	43.5	33.9	**9.2**	30.1
A2NET [L. Yang et al. 2020]	RGB+Flow	58.6	54.1	45.5	32.5	17.2	41.6	43.6	28.7	3.7	27.8
GTAD [M. Xu et al. 2020]	RGB+Flow	66.4	60.4	51.6	37.6	22.9	47.8	50.4	34.6	9.0	34.1
BSN++ [H. Su et al. 2021]	RGB+Flow	59.9	49.5	41.3	31.9	22.8	41.1	51.3	**35.7**	8.3	**34.9**
AFSD [C. Lin et al. 2021]	RGB+Flow	**67.3**	**62.4**	**55.5**	**43.7**	**31.1**	**52.0**	**52.4**	35.3	6.5	34.4
TALNET [Chao et al. 2018]	RGB	42.6	–	31.9	–	14.2	–	–	–	–	–
A2NET [L. Yang et al. 2020]	RGB	45.0	40.5	31.3	19.9	10.0	29.3	39.6	25.7	2.8	24.8
Baseline-III	RGB	36.3	31.9	25.4	17.8	10.4	24.3	28.2	18.3	3.7	18.2
Ours	RGB	**50.8**	**44.1**	**35.8**	**25.7**	**15.7**	**34.5**	**44.0**	**27.0**	**5.1**	**27.3**

Table 5.5: **Comparison to state-of-the-art approaches on closed-set action localisation**. AVG denotes the average mAP in [0.3:0.1:0.7] on THUMOS14, and [0.5:0.05:0.95] on ActivityNet1.3. Baseline-III uses the same proposal detector as Ours, but uses CLIP with hand-crafted templates as proposal classifiers.

for K-400, we follow [L. Zhu and Y. Yang 2018; Perrett et al. 2021] to collect 24 testing categories, each containing 100 videos. During training, we sample 5 action categories (ways) from the above data, with 5 videos (shots) from each category, and use the remaining data for evaluation. To ensure the statistical significance of experiments, we conduct 200 trials with random samplings.

Table 5.3 presents the average TOP1 accuracy for the three datasets. Our method (with or without temporal modeling) clearly outperforms all previous methods by a significant margin, around 10% on HMDB-51 and K-400, indicating the superiority of our proposed idea for model adaptation.

5-Shot-$\mathcal{C}$-Way Setting. Here, we further consider a more challenging scenario, scaling the problem up to classifying all categories in the dataset with only 5 samples per category, for example, $\mathcal{C}_{\text{ALL}} = 400$ for K-400, $\mathcal{C}_{\text{ALL}} = 101$ for UCF-101. Specifically, on each dataset, we sample 5 videos (shots) from the training set for each action category, to form the few-shot training data, and then measure performance on the corresponding standard testing set.

For this experiment setting, we conduct 10 random sampling rounds, and record the average TOP1 accuracy in Table 5.3. Comparing to the 5-way classification, the $\mathcal{C}$-way setting is clearly more challenging, our model (with or without temporal modeling) still demonstrates promising results. While comparing to the Baseline-I, our performance gains on UCF-101 and HMDB-51 are around 15%.

Open-set Action Recognition

In this section, we evaluate on open-set action recognition, with videos in the training set and validation set from different categories, *i.e.* $\mathcal{C}_{\text{train}} \cap \mathcal{C}_{\text{val}} = \emptyset$. Specifically, we split the K-700 dataset into two parts, with $\mathcal{C}_{\text{train}} = 400$ categories for training, and the remaining $\mathcal{C}_{\text{val}} = 300$ categories for evaluation. For more details on the dataset splits, please refer to the appendices.

As a baseline, we evaluate the CLIP model with handcrafted prompt templates. As reported in Table 5.4, our model achieves 6.0% improvements on TOP1 accuracy over the Baseline-I, showing the effectiveness of prompt learning for open-set recognition. Interestingly, the number of learnable prompt vectors does not make a difference, and adding temporal modeling diminishes the performance gain. We conjecture this is because the additional Transformer layer will specialise on the training set, thus harming the generalisation towards unseen action categories.

Conclusion & Discussion

Among all the benchmarks on action recognition, we have demonstrated the effectiveness of prompt learning and temporal modeling. For closed-set action recognition, even without temporal modeling, learnable prompts clearly surpass the handcrafted ones, and linear probe settings. While comparing to state-of-the-art approaches, despite training *far fewer* parameters, our model still demonstrates competitive performance on all benchmarks. For few-shot action recognition with limited number of training samples, model adaptation through prompt learning really shines, outperforming all previous methods significantly. Lastly, for open-set scenarios, textual prompts enable to make better use of the abundant training data, and further improve the generalisation beyond the seen actions.

5.4.3 Action Localisation

Datasets & Metrics. In this section, we conduct experiments on two localisation datasets. **THUMOS14** [Y.-G. Jiang et al. 2014] covers 413 untrimmed sports videos of 20 action categories, with an average of 15 action instances per video. The standard split is 200 training videos and 213 validation videos. **ActivityNet1.3** [Heilbron et al. 2015] has around 20k untrimmed videos of 200 action categories, each video contains an average of 1.5 action instances. The standard

split is $10,024$ training videos and $4,926$ validation videos. For evaluation metrics, we follow the conventional choices and use mean Average Precision (mAP) at different IoU thresholds. On THUMOS14, we report the mAP at the IoU set $[0.3 : 0.1 : 0.7]$; as for ActivityNet1.3, the IoU set is $[0.5 : 0.05 : 0.95]$.

Closed-set Action Localisation

Closed-set action localisation refers to the common setting, where the model is trained and evaluated on videos of the same action categories, *i.e.* $\mathcal{C}_{\text{train}} = \mathcal{C}_{\text{val}}$. For a fair comparison, we use the same dataset splits as in the literature.

Table 5.5 reports the comparison results. As a baseline, we adopt the same first-stage proposal detector, but use the original CLIP with handcrafted templates ("this is a photo of $[\cdot]$") for the second-stage proposal classifier. On both datasets, our model significantly outperforms the Baseline-III, again confirming the effectiveness of prompt learning and temporal modeling. While comparing with other existing methods that use pre-trained RGB stream, our method also demonstrates superior performance, with around 5.2% and 2.5% gains on average mAP respectively.

Open-set Action Localisation

In this section, we evaluate on the open-set scenario, *i.e.* action categories for training and testing are disjoint. As we are not aware of any existing benchmarks on this challenging problem, we initiate two evaluation settings on THUMOS14 and ActivityNet1.3: one is to train with 75% action categories and test on the left 25% action categories; the other is to train with 50% categories and test on the left 50% categories. To ensure statistical significance, we conduct 10 random samplings to split categories for each setting.

Table 5.6 shows the mean performance. In two-stage localisation, as proposals are class-agnostic, the key is the proposal classifier. Similar to closed-set localisation, for comparison, we implement the same baseline, which uses the same proposal detector as our model, but classifies action proposals using original CLIP with handcrafted prompts. In both settings, our model shows superior performance than the Baseline-III. However, when comparing with the closed-set evaluation, the performance of open-set drops dramatically. Note that, such performance drop now comes from two sources: one is the recall drop from the first-stage class-agnostic action proposals, because for open-set settings, there exist certain differences in

		THUMOS14						ActivityNet1.3			
Method	Train	0.3	0.4	0.5	0.6	0.7	AVG	0.5	0.75	0.95	AVG
Baseline-III	75%	33.0	25.5	18.3	11.6	5.7	18.8	35.6	20.4	2.1	20.2
Ours	75%	**39.7**	**31.6**	**23.0**	**14.9**	**7.5**	**23.3**	**37.6**	**22.9**	**3.8**	**23.1**
Baseline-III	50%	27.2	21.3	15.3	9.7	4.8	15.7	28.0	16.4	1.2	16.0
Ours	50%	**37.2**	**29.6**	**21.6**	**14.0**	**7.2**	**21.9**	**32.0**	**19.3**	**2.9**	**19.6**

Table 5.6: **Results of open-set action localisation**. Baseline-III uses the same proposal detector as Ours, but uses CLIP with handcrafted templates as the proposal classifier. Our model is trained on 75% (or 50%) action categories and tested on the remaining 25% (or 50%) action categories.

the boundary distribution of training and testing proposals; the other comes from the second-stage classification errors. In the appendices, we provide a full ablation study to show the performance degradation under the open-set scenario.

5.4.4 Text-Video Retrieval

Datasets & Metrics. In this section, we evaluate on three large-scale text-video retrieval datasets. **LSMDC** [A. Rohrbach et al. 2017] covers 118,081 videos of 2 to 30 seconds. We train on 7,408 validation videos, and evaluate on 1,000 independent test videos. **MSRVTT** [J. Xu et al. 2016] has 10,000 videos and 200,000 captions. We train on the 'Training-9K' split [Gabeur et al. 2020], and test with the 'test 1k-A' [Y. Yu et al. 2018] containing 1,000 clip-text pairs. **SMIT** [Monfort et al. 2021a] (Spoken Moments) contains over 500k videos randomly chosen from the M-MiT training set [Monfort et al. 2021b], and 10k validation videos. Each video contains at least one text description. For evaluation, following the literature, we report average recall at K (R@K). Due to the space limitation, we refer the readers for the full Table with median rank (MdR).

Results. Table 5.7 presents the results on three benchmarks. Note that, in these experiments, we only employ 8 learnable prompt vectors, *i.e.* [4+X+4]. This is because the text encoder from pre-trained CLIP takes limited number of textual tokens up to 77, whereas the text query for retrieval tends to be long. And for the cases where the tokenised text query is longer than the maximum supported length of CLIP, we truncate the sequence to fit our specified pattern.

While comparing with the Baseline-IV that denotes the results from the original CLIP model with naïvely-encoded text queries, our proposed prompt learning and temporal modeling module demonstrate clear benefits on all benchmarks. Com-

Method	E2E	MSRVTT (9K) R@1	MSRVTT (9K) R@5	LSMDC R@1	LSMDC R@5	SMIT R@1	SMIT R@5
CE [Yang Liu et al. 2019]	✗	21.7	51.8	12.4	28.5	–	–
MMT [Gabeur et al. 2020]	✗	24.6	54.0	13.2	29.2	–	–
TT-CE+ [Croitoru et al. 2021]	✗	29.6	61.6	**17.2**	**36.5**	–	–
Baseline-IV	✗	31.2	53.7	11.3	22.7	39.3	62.8
Ours ([4+X+4])	✗	**36.7**	**64.6**	13.4	29.5	**66.6**	**87.8**
Frozen [Bain et al. 2021]	✓	31.0	59.5	15.0	30.8	–	–
CLIP4Clip [H. Luo et al. 2021]	✓	44.5	71.4	22.6	41.0	–	–

Table 5.7: **Results of text-video retrieval**. Baseline-IV refers to the original CLIP model with text query naïvely encoded, *i.e.* without using any prompt. E2E denotes if the model has been trained end-to-end.

paring with the existing approaches that are specifically targeting the text-video retrieval problem, our proposed method still performs competitively, although it only requires to optimise a few prompt vectors, along with two Transformer layers on the considered retrieval datasets.

5.5 Conclusion

In this paper, building on image-based visual-language models, *e.g.* CLIP, we propose to learn task-specific prompt vectors for *efficient* and *lightweight* model adaptation. We evaluate the proposed idea on 9 popular benchmarks of major video understanding tasks: action recognition, localisation, and text-video retrieval. Thorough comparisons and ablation studies are conducted to analyse the critical components, at different scenarios: closed-set, few-shot, and open-set. In the closed-set scenario, despite training only a small number of free parameters, we achieve competitive performance to the modern state-of-the-art methods. In few-shot and open-set scenarios, we significantly outperform existing methods across all tasks, sometimes by over 10%.

Limitation and Broader Impact. Our proposed idea relies on the visual-language model pre-trained on the large-scale image alt-text data, which could potentially incur two limitations: *First*, bias in the web data, *Second*, as temporal modeling is only used on top of visual features, it may fail to model fine-grained motions. As future work, we expect better visual-language models to be trained, further improving model adaptation by proposed prompt learning; and given more compute available, end-to-end finetuning with all data combined will further boost

the performance.

Appendices

Appendices for this chapter can be found in the online version of the paper. [1]

Statement of authorship

A statement of authorship for this paper is provided in Appendix A.

[1] https://ju-chen.github.io/efficient-prompt/

Chapter 6

Turbo Training with Token Dropout

The paper has been accepted for publication at the British Machine Vision Conference (BMVC), 2022.

Turbo Training with Token Dropout

Tengda Han[1] Weidi Xie[1,2] Andrew Zisserman[1]

[1]Visual Geometry Group, University of Oxford

[2]Shanghai Jiao Tong University

https://www.robots.ox.ac.uk/~vgg/research/turbo/

Abstract

The objective of this paper is an efficient training method for video tasks. We make three contributions: (1) We propose Turbo training, a simple and versatile training paradigm for Transformers on multiple video tasks. (2) We illustrate the advantages of Turbo training on action classification, video-language representation learning, and long-video activity classification, showing that Turbo training can largely maintain competitive performance while achieving almost 4× speed-up and significantly less memory consumption. (3) Turbo training enables long-schedule video-language training and end-to-end long-video training, delivering competitive or superior performance than previous works, which were infeasible to train under limited resources.

6.1 Introduction

The extra temporal axis of video, compared to a static image, can capture rich information such as long term activities and stories. However, it also brings a few orders of magnitude of more data with the concomitant costs in processing and memory requirements. This, together with the long training schedules required to train networks on video data, has tremendously slowed research progress in video understanding and has raised the bar for researchers to make contributions in this area, particularly with limited resources. Furthermore, training consumes a significant amount of energy and hardware, with detrimental consequences for the natural environment.

In this paper, we propose a computation-efficient paradigm for training Transformers on video tasks, termed as **Turbo training**, which only needs to process sparsely sampled visual tokens. It is based on a multi-task training that jointly optimises two objectives: a masked region autoencoder loss, and the standard training loss for the downstream task of interest, for example, cross-entropy for action recognition, or contrastive loss for visual-language representation learning.

Turbo training is possible because the visual data contains redundancy. As observed in the recent self-supervised pre-training work, such as MAE [K. He et al. 2022] for images and [Tong et al. 2022; Feichtenhofer et al. 2022] for videos, it is sufficient to use only a small fraction of the input visual tokens to reconstruct the visual signal. For example, in images, 75% of the visual tokens can be dropped, whilst in videos, the dropout can be even more aggressive, and 90% of the spatio-temporal tokens can be dropped while still being able to learn a strong representation using self-supervision. In Turbo training, we adopt a similar intuition and target efficient supervised training for video understanding. In short: **given the high redundancy in videos, training a Transformer with sparsely sampled visual tokens is sufficient for the downstream tasks of interest**. This is beneficial for video understanding tasks in two ways: *first, training efficiency*, it dramatically reduces the computational cost for training transformers; *second, memory requirements*, videos of longer length can be loaded and directly trained on, leading to improved performance and new abilities, *i.e.*, end-to-end training the transformer models.

To summarise, we make the following contributions: (1) We propose Turbo

training, a new paradigm for training Transformers on video understanding tasks, that jointly optimises a masked autoencoding and standard training loss for the downstream task of interest. (2) We illustrate the advantages of Turbo training on three video understanding tasks, including action classification, video-language training, and long-video activity classification, showing that training on sparsely sampled video patches can largely maintain competitive performance on most of the video tasks of interest, while achieving almost 4× speed-up and significant reduction in memory consumption. As a consequence, (3) Turbo training enables long-schedule video-language representation learning and end-to-end long-video activity classification, delivering competitive or superior performance than previous methods.

6.2 Related Works

Masked Autoencoding has been used in the natural language processing community with great success to pre-train language models with a fill-in-the-blank task; specifically, the model is trained to use the context words surrounding a masked token to try to predict what the masked word should be, a typical example would be BERT [Devlin et al. 2018]. In fact, a similar idea has also been investigated in computer vision for self-supervised learning in the seminal work of context autoencoder [Pathak et al. 2016], where the authors proposed to learn a visual representation by training a ConvNet to inpaint the missing region of an image. However, due to the large effective spatial receptive fields of ConvNets, severe boundary artefacts weakened the learnt representation. Recently, with the advent of the Vision Transformers [Dosovitskiy et al. 2020], that do not suffer from the same spatial receptive field problem, masked autoencoding has become popular again as an effective method to pre-train large-scale Vision Transformers. Examples include MAE [K. He et al. 2022], SimMiM [Zhenda et al. 2022], and BEiT [Bao et al. 2022] for images, and [Feichtenhofer et al. 2022; Tong et al. 2022; Agrim Gupta et al. 2022] for videos.

Efficient Deep Neural Networks. Since the resurgence of deep neural networks, reducing the computational cost for training or inference has always considered as an important topic. For instance: in image classification, novel architectures are regularly developed, Binarised Neural Networks [Courbariaux et

al. 2016; Rastegari et al. 2016], MobileNet [Howard et al. 2017], ShuffleNet [X. Zhang et al. 2018], EfficientNet [M. Tan and Le 2019]; and in object detection, single-stage detectors have drastically improved the efficiency of object detection, *e.g.*, YOLO [Redmon et al. 2016], SSD [W. Liu et al. 2016], RetinaNet [T.-Y. Lin et al. 2017], CornerNet [Law and J. Deng 2018], *etc.* When processing videos, the efficiency becomes even more critical. In the literature, approaches have been proposed to intelligently sample the video data, by exploiting the relatively cheap audio modality in videos [Korbar et al. 2019], to exploit variable mini-batch shapes for fast training [C.-Y. Wu et al. 2020], or to directly train on compressed videos without decoding [Z. Wu et al. 2018].

Efficient Transformers. Transformer model architectures have gained immense interest due to their effectiveness on a range of tasks, including computer vision, natural language processing, reinforcement learning, *etc.* The self-attention mechanism is a critical design in Transformer that enables each input token to explicitly build relationships with others. In order to improve the general efficiency, a rich set of ideas have been explored to reduce the complexity for computing the self-attention affinity matrix, for example, to develop recurrence in the Transformer (Transformer-XL [Dai et al. 2019]), to reduce the computation consumption by only exploiting Transformer decoders with a controllable number of queries (Perceiver [Jaegle et al. 2021]), to use learnable attention pooling (Routing Transformer [Roy et al. 2020], Sinkhorn Transformer [Tay et al. 2020], Reformer [Kitaev et al. 2020]) to rely on low rank matrix decomposition of the affinity matrix from self-attention (Linformer [S. Wang et al. 2020], MotionFormer [Patrick et al. 2021b], Performer [Choromanski et al. 2021], Linear Transformer [Katharopoulos et al. 2020]), to exploit sparse representation (Product Key Memory [Lample et al. 2019]) for the affinity matrix. In this paper, our proposal is orthogonal to the above mentioned techniques, specifically, motivated by the high redundancy in visual signals, we aim to train the Transformer on videos with only sparsely sampled visual patches.

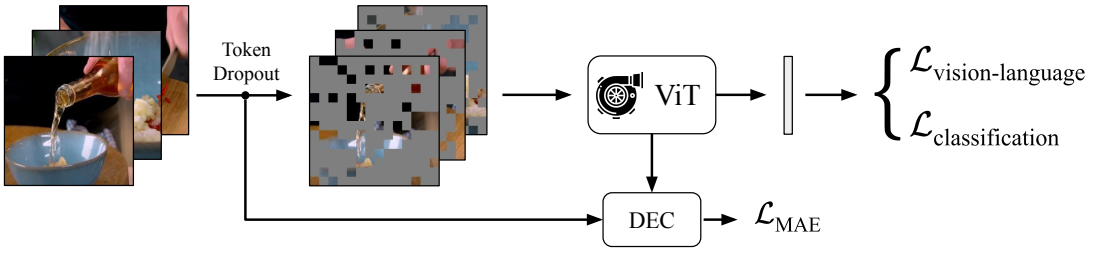


Figure 6.1: Illustration of Turbo training. We train a Video Transformer model by only using sparsely sampled visual tokens, for faster training and reduced computational cost. To achieve this goal, we exploit a multi-task training scheme, with one task being partial masked autoencoding, and the other being the standard supervised loss for the downstream task of interest, for example, contrastive learning for visual-language pre-training, or cross-entropy for action classification.

6.3 Method

We start by presenting an overview of Vision Transformers in Section 6.3.1, and discuss their pros and cons for video understanding tasks. In Section 6.3.2, we introduce the **Turbo** training, and describe its application for efficiently training the Vision Transformer on videos.

6.3.1 Vision Transformers for Videos

Generally speaking, given a video with T frames, *i.e.*, $\mathcal{V} = \{I_1, I_2, \ldots, I_T\}$, with $I_k \in \mathbb{R}^{H \times W \times 3}$, as input to a Vision Transformer (ViT), the video is first split into non-overlapping spatial-temporal patches, and projected with linear layers to get a sequence of token vectors. After appending a learnable 'CLS' token ($v_{\texttt{CLS}}$), the resulting vector sequence is then processed with series of transformer encoder layers, *i.e.*, $\mathbf{z} = \Phi_{\text{ViT}}([v_{\texttt{CLS}}, v_1, ..., v_n]+\texttt{PE})$, where v_i denotes one of the $n = n_t \cdot n_h \cdot n_w$ spatio-temporal patches of size $t \times h \times w$, with $n_t = \lfloor T/t \rfloor, n_h = \lfloor H/h \rfloor, n_w = \lfloor W/w \rfloor$, and PE refers to the spatio-temporal positional encodings.

With the self-attention operations along *all* the spatial-temporal patches, the benefit of applying ViT on videos is prominent – it can capture the long-term dependency in the videos. However, the self-attention also incurs the quadratic complexity, *i.e.*, $\mathcal{O}((n_t \cdot n_h \cdot n_w)^2)$, for scenarios when the input video is long (more than a few seconds), the memory consumption becomes infeasible. As a consequence, the majority of ViT-based architectures are still limited to only processing short video clips.

6.3.2 Turbo Training

In this paper, we introduce a **Turbo training** regime, that aims to exploit the redundancy in video data, and optimise a video Transformer for multi-task training with only sparsely sampled visual tokens. It requires optimizing for two objectives: (i) partial masked autoencoding, that acts as a proxy task to force the model to capture the spatio-temporal information to a full extent, with only the sparsely sampled video patches; and (ii) the default objective for the considered downstream task, that may refer to cross-entropy or contrastive learning.

In the following, we start by describing the partial masked autoencoder, and then detailing the use of Turbo training on various video understanding tasks, including: action classification, video-language training and long-video activity classification.

Partial masked autoencoder (PMAE). We use a ViT as the visual encoder ($\Phi_{\text{ViT-enc}}(.)$), that consists of a number of Transformer encoder blocks, operating on the input visual tokens. Inspired by MAE [K. He et al. 2022], we also use a shallow decoder with much fewer Transformer blocks than the encoder ($\Phi_{\text{dec}}(.)$). After preparing the spatial-temporal patches, we define three operations for the embedded tokens: (1) masking the input visual tokens with a ratio m, which means only $N_i = n_t \cdot n_h \cdot n_w \cdot (1-m)$ tokens are fed into the visual encoder ($\Phi_{\text{ViT-enc}}(.)$); (2) reconstructing at a reconstruction ratio r, that means $N_r = n_t \cdot n_h \cdot n_w \cdot r$ tokens are the targets of the reconstruction tasks; (3) the unused $N_{\text{ignore}} = n_t \cdot n_h \cdot n_w \cdot (m-r)$ tokens are ignored in the training iteration. Note that a natural constraint $r \leq m$ holds, and the original MAE setting [K. He et al. 2022] can be regarded as the special case when $r = m$. This training regime significantly cuts downs the memory consumption, for example, the complexity for self-attention in the encoder layer is only $\mathcal{O}(N_i^2)$, compared to operating on the full token sequence ($\mathcal{O}((n_t \cdot n_h \cdot n_w)^2)$), while the decoder layer can be quite lightweight, with the complexity of self-attention being $\mathcal{O}((N_i+N_r)^2)$.

Action Classification. Here, we apply the Turbo training for Transformers on the video action classification task. In addition to the above-mentioned partial MAE loss ($\mathcal{L}_{\texttt{PMAE}}$), we adopt the standard cross-entropy loss on the short clips. Specifically, when given a labelled video sample $\{\mathcal{V}, y\}$, *i.e.*, a short video clip

with an action label, we take the '`CLS`' feature from the final layer of the visual encoder, then we pass the visual feature to a classifier, that is parametrised with a single MLP layer (denoted as $g(.)$), and trained with cross entropy loss. The overall objective for classification training is

$$\mathcal{L} = \lambda_{\texttt{CE}} \cdot \mathcal{L}_{\texttt{CE}}(g(z_{\texttt{CLS}}), y) + \mathcal{L}_{\texttt{PMAE}} \tag{6.1}$$

In practice we set the weight $\lambda_{\text{CE}} = 1/\log(\texttt{num\_classes})$ to balance two losses.

Visual-language Pre-training. Recently, visual-language representation learning has attracted growing interest from the community, due to its convenience of data curation procedure and remarkable performance on "zero-shot" generalisation on various image classification tasks [Radford et al. 2021; C. Jia et al. 2021]. However, the challenge for training the model lies in its requirement for a significant amount of compute and long training schedule. Turbo training can be applied to this scenario, to significantly speed up and reduce the memory cost. In addition to the sparse MAE loss ($\mathcal{L}_{\texttt{PMAE}}$), here we also adopt a standard InfoNCE loss (noise contrastive learning [van den Oord et al. 2018]) to learn the joint embedding for the visual and language streams.

Specifically, given a paired video-language sample $\{\mathcal{V}, y\}$, *e.g.*, a video clip and a sentence that describes the visual scene, we use a Transformer-based language model $\Psi_{\text{text}}(\cdot)$ to encode the sentence, and take the language '`CLS`' token as the representation for the entire sentence:

$$z_t = \Psi_{\text{text}}([w_{\texttt{CLS}}, w_1, \ldots, w_n] + \texttt{PE})$$

where w_i denotes word embedding after tokenisation, and `PE` denotes the positional embedding for the language. As for the visual stream, we use the same Transformer-based visual encoder ($\Phi_{\text{ViT-ENC}}(\cdot)$), as introduced above, and take the final visual '`CLS`' feature as the visual feature. The overall visual-language training loss is

$$\mathcal{L} = \lambda_{\texttt{NCE}} \cdot \mathcal{L}_{\texttt{NCE}} + \mathcal{L}_{\texttt{PMAE}} \tag{6.2}$$

with $\mathcal{L}_{\texttt{NCE}}$ denoting a bi-directional (visual-to-language & language-to-visual) In-

foNCE loss.

$$\mathcal{L}_{\texttt{NCE}} = -\frac{1}{2}\left(\log\frac{\exp\left(z_v \cdot z_t\right)}{\sum_t \exp\left(z_v \cdot z_t\right)} + \log\frac{\exp\left(z_v \cdot z_t\right)}{\sum_v \exp\left(z_v \cdot z_t\right)}\right)$$

In practice we set the weight $\lambda_{\texttt{NCE}} = 1/\log\left(\texttt{batch\_size}\right)$ to balance two losses.

Long-video activity classification. Turbo training enables to load longer video sequences for end-to-end training, which was a challenging factor for long-video tasks. By applying Turbo training, the method for long-video activity classification is largely simplified and is similar to the method of short-video action classification. Specifically, given a labelled long video sample $\{\mathcal{V}, y\}$, we still optimise Eq. 6.1 as the main objective, whilst applying a larger masking ratio on the input video $\mathcal{V}$ is necessary to reduce computational cost.

6.4 Experiments

In this section, we validate the effectiveness of Turbo training by experimenting on three different tasks: video action classification, video-language representation learning, and long-video activity classification.

6.4.1 Action Classification

Dataset. We conduct experiments on two datasets, **UCF101** [Soomro et al. 2012], containing 13k short video clips for 101 human actions, and **HMDB51** [Hildegard Kuehne et al. 2011], containing 7k short video clips for 51 human actions. We report the Top1 classification accuracy on this task.

Implementation. In accordance with the experiment setting as used in [Tong et al. 2022], for each short video clip we decode the video with 5 fps and randomly sample 16 video frames at 224×224 resolution as input, which approximately spans about 3.2 seconds in time. The video frames are passed to the $\Phi_{\text{ViT}}(\cdot)$ with a token dropout controlled by the mask ratio m, and we take the final-layer 'CLS' token as the video feature, which is further passed to a linear classifier and trained with cross-entropy (CE) loss. Also, we keep a light-weight MAE objective controlled by a reconstruction ratio r, that is to reconstruct some randomly selected $(r \times 100\%)$

space-time patches with a light weight ViT decoder. The overall objective function is in equation 6.1. We experiment with different combinations of the mask and reconstruction ratios $\{m, r\}$, to study their effects on turbo training. For more training details, we refer the reader to Appendix.

6.4.2 Video-Language Representation Learning

Dataset. For the experiments on video-language pre-training, we adopt the recent **HTM-AA** dataset, where the video and language description have been temporally auto-aligned with self-supervision [Han et al. 2022]. HTM-AA contains 250K videos sourced from the HowTo100M [Miech et al. 2019] dataset, with over 3.3M clip-sentence pairs. For evaluation of the learnt video-language representation, we benchmark on a temporal action alignment task on the **HTM-Align** dataset [Han et al. 2022]. Specifically, HTM-Align contains 80 long videos from YouTube with a total of 5K ASR sentences, and each visually alignable sentence has a **manually** labelled temporal segment. Note that, we only use this HTM-Align dataset for evaluation purposes, and all training is done on the automatically curated HTM-AA dataset, thus it can be seen as a 'zero-shot' evaluation on the HTM-Align.

Implementation. In accordance with [X. Lin et al. 2022], we choose MPNet [Song et al. 2020] as the language encoder. MPNet takes as input a sentence and outputs a textual feature vector. On the visual side, we use $\Phi_{\text{ViT}}(\cdot)$ and take the final-layer 'CLS' token as the video feature. To demonstrate the effectiveness of Turbo training for the Vision Transformer, we fix the weights of the language model, and only finetune the weights of the ViT architecture, initialized from VideoMAE [Tong et al. 2022] (pretrained on Kinetics-400 [Kay et al. 2017]). As in Sect. 6.4.1, the visual encoder takes as input a 16-frame short video clip extracted with 5 fps, and the language encoder takes the corresponding sentence associated with the video clip. Overall, we take a batch size of at most 32 clip-sentence pairs (depending on different masking ratios) for video-language training, with an InfoNCE loss. We refer the reader to Appendix for more training and evaluation details.

6.4.3 Long-video Activity Classification

Dataset. To demonstrate the effectiveness of exploiting Turbo training for end-to-end Transformer training on long video sequences, here we evaluate long-video activity classification on the Breakfast [Hilde Kuehne et al. 2014] and COIN [Y. Tang et al. 2019] datasets. In detail, **Breakfast** contains 1712 long videos for 10 cooking activities including 'making coffee' and 'frying egg'. **COIN** contains 11k long videos covering 180 general activities including 'car polishing' and 'assembling sofa'. The videos in these two datasets are often minute-long and untrimmed, containing procedural actions to complete the particular activity.

Implementation. In our experiment, we load $n = \{16, 32, 64\}$ frames as the video input, and train the Transformer end-to-end. Note that, such an end-to-end setting was not easily achievable in previous works due to the large memory footprint. In detail, at the training time, we randomly choose the start and end timestamp among the first and last 20% duration of the video, then uniformly take n frames in between. The rest of the architectural design is similar to that of Sect. 6.4.1. At inference time, we repeat sampling n frames for 10 times from each video, and average the prediction probability, resembling the multi-crop test that has been widely used in existing work [Simonyan and Zisserman 2014]. By default, we use a batch size of 16 videos and train the Transformer for 100 epochs on Breakfast and 50 epochs on COIN.

6.5 Results

Here, we start by showing the efficiency improvement from Turbo training on action classification (Sect. 6.5.1), then compare with existing works (Sect. 6.5.1). Next, we report the results for video-language representation learning (Sect. 6.5.2), followed by long-video activity classification (Sect. 6.5.3).

6.5.1 Turbo Training for Action Classification

Ablation Study: the speed and performance trade-off

In action classification, we investigate two of the hyperparameters of the PMAE, namely the mask ratio m and reconstruction ratio r. For the training, we use

Mask%	Recon%	GFLOPs	Acc (%)
0	- (w/o MAE)	180.6	94.8*
50	50 (MAE)	99.3	93.7
75	75 (MAE)	57.6	91.8
75	25 (PMAE)	45.9	91.6
90	90 (MAE)	35.2	89.7
90	10 (PMAE)	18.3	89.6

Train (m%,r%)	Inference m%	Acc (%)
90, 10	90	89.6
90, 10	50	90.1
90, 10	0	**90.3**
75, 25	75	91.6
75, 25	50	91.9
75, 25	0	**92.4**

Table 6.1: **Left**: Speed and performance trade-off on the UCF101 classification task. *: our implementation of [Tong et al. 2022] on UCF101. **Right**: Generalisation to different mask ratios at the inference time.

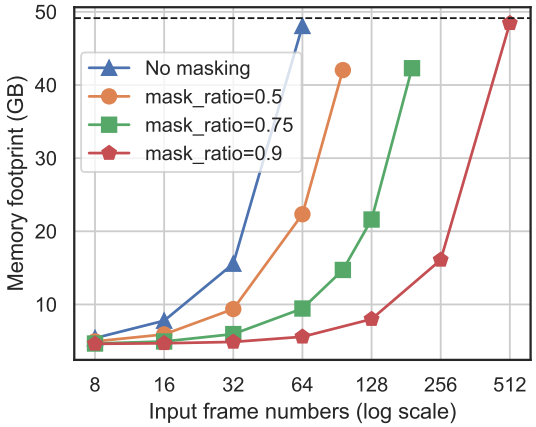


Figure 6.2: Memory footprint of training one video with different input frames.

ViT-B initialized with the VideoMAE weights, and then finetune the network on UCF101 for action classification.

Both high mask ratio and low reconstruction ratio can reduce the computation. In Table 6.1 (left), we compute the GFLOPs for the Transformer model during Turbo training. Note that GFLOPs contains both the classifier head for action classification and the decoder for patch reconstruction. It is evident that increasing the masking ratio can reduce the GFLOPs, *e.g.*, by applying a 50% masking ratio for action classification, the GFLOPs can be reduced by half at a cost of only 1% Top1 classification accuracy. Additionally, we find reducing the reconstruction ratio can also reduce the GFLOPs while incurring minimum effect to the final classification accuracy (91.8 to 91.6, and 89.7 to 89.6). In the following experiments, we use settings with $\{m, r\} = \{0.75, 0.25\}$ and $\{0.5, 0.5\}$, that demonstrate a balance between the performance and the compute operations. Figure 6.2 and 6.3 show the reduced memory footprints and the significant speed-up during training.

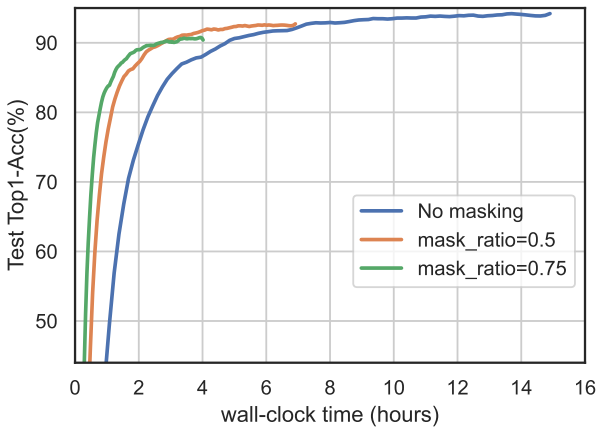


Figure 6.3: Training progress on UCF101 with a batch size of 16 for 100 epochs.

Generalisation to smaller masking ratio during inference. Here, we take the models trained with $\{m, r\} = \{0.9, 0.1\}$ and $\{0.75, 0.25\}$, and test it with different masking ratio at inference time. In Table 6.1 (right), we show that the model trained with high-rate token dropout can be used with a lower mask ratio at inference time. For example, inference with full patches ($m = 0$) achieves the best performance regardless of the training masking ratio. In the following experiments, we thus use $m = 0$ at inference by default.

Comparison to state-of-the-art

In Table 6.2, we compare to the existing approaches, which have been pre-trained on video or multimodal data in a self-supervised manner, and then end-to-end finetuned on the downstream tasks. **Note that**, the comparison here is by no means fair, due to the differences in architecture, training data, and resolutions. However, the key message in this comparison is that, even with only 50% of the visual tokens for finetuning, the model can still achieve competitive performance compared to all other models, while only needing half the training time as the original finetuning with all visual tokens. Additionally, with further self-supervised Turbo training on video-language data (in Sect. 6.5.2), the performance can be further boosted.

Finetune Method	Pretrain Method	Backbone	Pretrain Data	Modality	UCF	HMDB
Classic	CoCLR [Han et al. 2020b]	S3D	K400	V	87.9	54.6
Classic	CVRL [Qian et al. 2020]	Slow-R152	K600	V	93.6	69.4
Classic	ρ-BYOL [Feichtenhofer et al. 2021]	Slow-R50	K400	V	94.2	72.1
Classic	VideoMAE [Tong et al. 2022]	ViT-B	K400	V	**96.1**	73.3
Classic	MIL-NCE [Miech et al. 2020]	S3D-G	HTM	V+T	91.3	61.0
Classic	TAN [Han et al. 2022]	S3D-G	HTM-AA	V+T	92.0	-
Classic	XDC [Alwassel et al. 2020]	R(2+1)D-18	IG65M	V+A	94.2	67.1
Classic	GDT [Patrick et al. 2021a]	R(2+1)D-18	IG65M	V+A	95.2	72.8
Turbo$_{m=0.5,r=0.5}$	VideoMAE [Tong et al. 2022]	ViT-B	K400	V	94.4	71.8
Turbo$_{m=0.5,r=0.5}$	NCE	ViT-B	HTM-AA	V+T	95.7	**73.6**

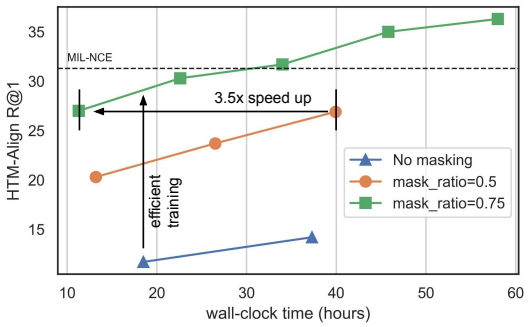
Table 6.2: Comparing with recent self-supervised methods by finetuning on UCF101 and HMDB51 action classification. We apply Turbo training for finetuning, 'Classic' means traditional finetuning method without any token dropout or not applicable to token dropout. We use $\{m, r\} = \{0.5, 0.5\}$ for our results.

6.5.2 Turbo Training for Video-Language Representation Learning

In this section, we experiment with video-language turbo training on the HTM-AA dataset. In detail, we consider three settings: no masking, $\{m, r\} = \{0.5, 0.5\}$, and $\{0.75, 0.25\}$. For each setting, we use the same implementation and the maximum number of video sequences a single NVidia A40 GPU can handle. We measure the quality of the video-language representation by monitoring the temporal action alignment performance on the HTM-Align dataset [Han et al. 2022], *i.e.*, training wall-clock time vs. the temporal alignment performance. In detail, we extract the per-second visual features for the long videos in HTM-Align, then use the text embeddings to retrieve the corresponding visual moment in the time axis, recall is 1.0 if the predicted timestamp fall into the ground-truth temporal segment. We report Recall@1 (R@1) only on alignable sentences as in [Han et al. 2022].

As shown in Figure 6.3 (left), it is clear that a higher mask ratio significantly increases the training efficiency. For example, after being trained for 18h, the 0.75 mask ratio achieves 28% R@1, whereas the baseline setup only gets 11% R@1. Additionally, to get the same performance, the $m = 0.75$ setting gives a 3.5× speed-up comparing with $m = 0.5$ (reaching 27% R@1 requires 11.3 hours and 40 hours respectively). For the baseline setting without Turbo training, it is not feasible to achieve a comparable performance under a single GPU training budget, showing that Turbo training indeed leads to improved performance and new abilities for existing Transformer models to finetune on video tasks in an end-to-end manner.

In Table 6.3 (right), we compare with previous methods on HTM-Align dataset.



Method	Pretrain Data	R@1
CLIP (ViT-B-32) [Radford et al. 2021]	YFCC	16.8
MIL-NCE [Miech et al. 2020]	HTM	31.3
TAN [Han et al. 2022]	HTM	49.4
Turbo (m=0.75)	HTM-AA	**36.7**

Table 6.3: **Left:** progress of vision-language pretraining monitor by R@1 on HTM-Align dataset. **Right:** zero-shot text-visual alignment results on HTM-Align dataset. We adopt the same setting as [Han et al. 2022] and report Recall@1 as the metric.

Note that our representation is not comparable with TAN [Han et al. 2022], in which their model contains temporal transformers modelling the surrounding minute-long context, while our model only extracts per-second features for this alignment task. Importantly, our Turbo training method surpasses MIL-NCE representation (36.7% vs. 31.3%), which was trained using 64 TPUs and a long training schedule.

6.5.3 Turbo Training for Long-video Activity Classification

Turbo training enables the end-to-end training of Transformers on long-video tasks, that used to be a critical bottleneck. We compare three settings: **(1)** 16-frame input with $\{m, r\} = \{0.5, 0.5\}$ (denoted as F_{16}), **(2)** 32-frame input with $\{m, r\} = \{0.75, 0.25\}$ (denoted as F_{32}), and **(3)** 64-frame input with $\{m, r\} = \{0.875, 0.125\}$ (denoted as F_{64}). Each of them has the same number of *visible* patches to the encoder but with different temporal extent. For frame sampling, we first sample the starting and ending positions from the beginning and the last 20% of the video, then we uniformly sample frames within this segment.

Effect of loading longer sequences. In Table 6.4, increasing the frame numbers from 16 to 32 (row B vs. C) shows the longer temporal extent is beneficial for activity classification (e.g. 91.3 vs. 88.2 on Breakfast), we conjecture a denser temporal sampling has a higher chance to sample the characteristic moment in the long video. However, further increasing the frame numbers to 64 does not help the activity classification task. It might be because the high token masking ratio ($m = 0.875$) could frequently miss the key objects or actions in the video frame.

Temporal Method	Clip Model	Pretrain Supervision	Pretrain Data	BF	COIN
Timeception [Hussein et al. 2019a]	3D-ResNet	[✓] action labels	K400	71.3	-
VideoGraph [Hussein et al. 2019b]	I3D	[✓] action labels	K400	69.5	-
GHRM [J. Zhou et al. 2021]	I3D	[✓] action labels	K400	75.5	-
Transformer [X. Lin et al. 2022]	TimeSformer	[✓] action labels	K400	81.1	83.5
Transformer [X. Lin et al. 2022]	TimeSformer	[✗] k-mean on ASR	HTM	81.4	85.3
Transformer [X. Lin et al. 2022]	TimeSformer	[✗] Dist. Sup	HTM+WikiHow	89.9	**88.9**
[A] **Turbo** F_{32}	N/A (end-to-end)	[✗] VideoMAE	K400	86.8	82.3
[B] **Turbo** F_{16}	N/A (end-to-end)	[✗] NCE	HTM-AA	88.2	81.2
[C] **Turbo** F_{32}	N/A (end-to-end)	[✗] NCE	HTM-AA	**91.3**	87.5
[D] **Turbo** F_{64}	N/A (end-to-end)	[✗] NCE	HTM-AA	87.3	86.8

Table 6.4: Procedural activity classification on the Breakfast (BF) and COIN dataset. [✓] denotes supervised training and [✗] denotes training without manual labelling.

Comparison with other works. When learning from 32 frames on the Breakfast dataset end-to-end, our simple one-stage method outperforms previous methods (91.3 vs. 89.9) which typically require a two-stage setting: extracting short-clip features, then training temporal models on these pre-extracted features. On the COIN dataset, our one-stage method achieves comparable results (87.5 vs. 88.9). We conjecture a longer temporal context with smaller masking ratio could further improve the performance on uncurrated natural videos like COIN, which inevitably demands higher hardware requirements. Comparing row C and row A further shows the effectiveness of our vision-language training from Sect. 6.5.2.

6.6 Conclusion

We propose a simple and versatile Turbo training paradigm for Vision Transformers. We show that it is applicable to multiple video tasks including action classification, video-language training, and long-video activity classification, and can achieve superior performance, while significantly reducing the required computation and memory cost. Turbo training has demonstrated the ability to lower the resource requirements and enable end-to-end Transformer training for long videos.

Acknowledgement

This research is funded by EPSRC Programme Grant VisualAI EP/T028572/1, a Royal Society Research Professorship RP\R1\191132, and a Google-DeepMind Scholarship. We thank Charig Yang and Chuhan Zhang for proofreading and discussions.

Appendices

Appendices for this chapter can be found in the online version of the paper. [1]

Statement of authorship

A statement of authorship for this paper is provided in Appendix A.

[1] https://www.robots.ox.ac.uk/~vgg/research/turbo/

Part III

Self-supervised Learning for Long-term Videos

Chapter 7

Temporal Alignment Networks for Long-term Video

The paper has been accepted for publication at the Conference on Computer Vision and Pattern Recognition (CVPR), 2022.

Temporal Alignment Networks for Long-term Video

Tengda Han[1] Weidi Xie[1,2] Andrew Zisserman[1]

[1]Visual Geometry Group, University of Oxford

[2]Shanghai Jiao Tong University

https://www.robots.ox.ac.uk/~vgg/research/tan/

Abstract

The objective of this paper is a temporal alignment network that ingests long term video sequences, and associated text sentences, in order to: (1) determine if a sentence is alignable with the video; and (2) if it is alignable, then determine its alignment. The challenge is to train such networks from large-scale datasets, such as HowTo100M, where the associated text sentences have significant noise, and are only weakly aligned when relevant.

Apart from proposing the alignment network, we also make four contributions: (i) we describe a novel co-training method that enables to denoise and train on raw instructional videos without using manual annotation, despite the considerable noise; (ii) to benchmark the alignment performance, we manually curate a 10-hour subset of HowTo100M, totalling 80 videos, with sparse temporal descriptions. Our proposed model, trained on HowTo100M, outperforms strong baselines (CLIP, MIL-NCE) on this alignment dataset by a significant margin; (iii) we apply the trained model in the zero-shot settings to multiple downstream video understanding tasks and achieve state-of-the-art results, including text-video retrieval on YouCook2, and weakly supervised video action segmentation on Breakfast-Action; (iv) we use the automatically-aligned HowTo100M annotations for end-to-end finetuning of the backbone model, and obtain improved performance on downstream action recognition tasks.

7.1 Introduction

The recent CLIP and ALIGN papers [Radford et al. 2021; C. Jia et al. 2021] have demonstrated that a combination of large scale paired image-caption data, and a simple noise contrastive learning loss can be used to learn powerful image-text embeddings from scratch. The image-caption data can be crawled from the internet at scale, for example from image alt-text, and the resulting embeddings demonstrate strong "zero-shot" generalization abilities. In the video domain, there also exists large-scale sources of text supervision, *e.g.* narrated instructional videos such as the HowTo100M [Miech et al. 2019] dataset, where demonstrators explain their actions while performing a complex task. The narrations are unconstrained and can be combinatorially complex, including information on "what", "where" and "when", such as the actions, the objects, human-object interactions, *etc.*

However, these instructional videos pose additional fundamental challenges over the image-caption scenario due to the temporal *alignment* problem (illustrated in Figure 7.1): (i) the demonstrator often makes statements that are unrelated to the visual signal, such as describing food taste or explaining the consequence of actions. These texts are *not visually alignable*. (ii) the demonstrator might explain their action before or after performing it, and their statements often *do not follow the same order* as their actions, resulting in the text and visual entities being asynchronous. These texts are *not temporally aligned* to the visual signal. Additionally, unlike spatial segmentation in images, where objects boundaries are often formed by a discontinuity between regions with strong gradients, temporal actions in videos are often continuous, making it difficult to clearly define the start and end points for the temporal interval. Last but not the least, there is additional noise coming from the imperfect Automatic Speech Recognition (ASR) systems on the spoken narrations. Note that the image-caption data does not face these problems since captions are provided by human annotators for that image; although they may be incomplete, there is no temporal alignment issue.

The extent of these alignment challenges is significant [Miech et al. 2019; Miech et al. 2020]. In 10 hours of instructional videos (sourced from HowTo100M) that we annotated for this work, only 30% of the narration sentences are visually alignable, and only 15% are naturally well-aligned. This means that the demonstrator is describing their action synchronously with the video only 15% of the time. If the

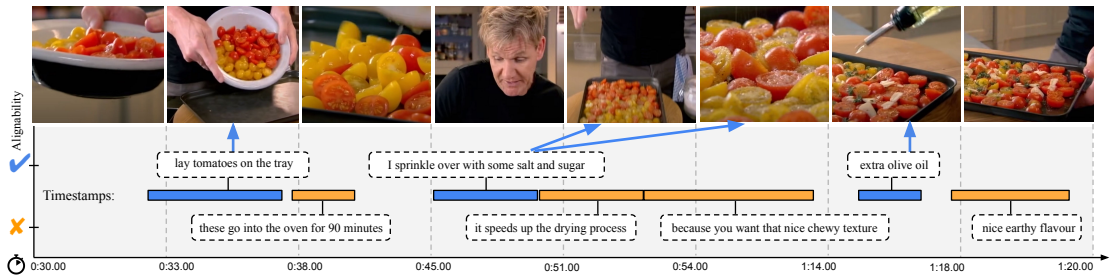


Figure 7.1: An example of visual-textual mis-alignment in a raw instructional video. The presenter's narration can be not visually relevant at all, *e.g.* describing a flavor; or asynchronous with visual content by a time difference. The ✓ and ✗ indicate visually alignable and non-alignable text, respectively (by human judgement). The colored bar shows the start-end timestamp of narration. Example from https://www.youtube.com/watch?v=M8OGXmLTTiI?t=30.

alignment issues are resolved then the benefits of learning from such narrated instruction videos can potentially be substantial: with the extra time axis alignment, models can be trained to deal with fine-grained tasks, and predict temporal action localization and segmentation.

In this paper, we tackle the sentence-to-video temporal alignment problem, and propose a Temporal Alignment Network (TAN) that ingests a video sequence and its associated narrative sentences, attends to a large temporal context in both, and is able to: (1) determine if a sentence is alignable with the video; and (2) if it is alignable, then determine its temporal alignment. Given all the challenges described above, training such a network on raw instructional videos, *e.g.*, HowTo100M, is clearly a non-trivial task. To this end, we propose a novel method for denoising, by co-training TAN with an auxiliary dual encoder network. By design, these two networks use complementary architectures: TAN iteratively attends to temporal context from both visual and textual modalities, establishing accurate alignment for sentences that are alignable; while the dual encoder processes visual and textual modalities independently, which enables it to spot unalignable sentences at ease, *e.g.*, sentences that emit low alignment score to all frames within the video. The output from these two networks can be treated as two different views for alignment, and their *mutual agreements* are adopted for co-training.

In addition to introducing the model and training methodology, we make the following contributions: (1) We manually annotate an 80-video subset of HowTo100M, named **HTM-Align**, by assigning the visually related sentences to their corresponding timestamps and annotating visually unrelated ones. This aligned subset is used to evaluate the model's performance and is released publicly;

(2) We train the model on the HowTo100M dataset, and demonstrate a significant improvement in alignment over prior work (MIL-NCE approach of [Miech et al. 2020] in particular); (3) We apply the trained model in both the zero-shot and fine-tuned settings to multiple downstream video tasks and achieve state of the art results on both settings. This includes text-video retrieval on YouCook2 [L. Zhou et al. 2018a] and weakly supervised video action segmentation on Breakfast-Action [Hilde Kuehne et al. 2014]; (4) We use the automatically-aligned HowTo100M annotations to finetune the *backbone model*, and observe improved performance on downstream action classification tasks.

7.2 Related Work

Joint Visual-Textual Learning has a long history in computer vision. As examples, early work from Mori et al. [Y. Mori et al. 1999] explored the connection between image and words in paired text documents, and [Weston et al. 2011] learnt a joint image-text embedding for the case of class name annotations. Recent works like CLIP [Radford et al. 2021] and ALIGN [C. Jia et al. 2021] show that large-scale paired image-caption data combined with a simple noise contrastive learning loss is able to learn a powerful visual representation. In video domains, this is also true, as shown by MIL-NCE [Miech et al. 2020], ALBEF [Junnan Li et al. 2021], and VideoClip [Hu Xu et al. 2021].

Visual-Textual Retrieval learns a joint embedding space for both vision and language, either using a dual encoder [Bain et al. 2021; Gong et al. 2014b; Gong et al. 2014a; Klein et al. 2015; Pan et al. 2016; Plummer et al. 2017; Dong et al. 2019; Miech et al. 2019; Radford et al. 2021; C. Jia et al. 2021], where visual and textual inputs are independently encoded, or a joint encoder, constructed with multimodal Transformers [H. Tan and Bansal 2019; Lu et al. 2019; W. Su et al. 2019; Lu et al. 2020; Y.-C. Chen et al. 2020; G. Li et al. 2020; L. Zhou et al. 2020], where vision and text inputs are fed into the cross-modal attention to compute the similarity. Despite being more accurate, the incurred computation of the joint encoder limits its use for large-scale retrieval systems. In [Miech et al. 2021], the authors propose to speed up the process by only using the joint encoder for re-ranking. In this work, we also use both joint and dual encoders, but for a different

purpose – to exploit their complementary information for co-training.

Visual-Textual Alignment aims to temporally assign words or sentences to the corresponding video segments. A similar task is weakly-supervised action segmentation that tries to delineate the video segments corresponding to a given action list [Bojanowski et al. 2014; Bojanowski et al. 2015; Hilde Kuehne et al. 2017; Ding and C. Xu 2018; D.-A. Huang et al. 2016; Richard et al. 2018; C.-Y. Chang et al. 2019; Zhukov et al. 2019; Jun Li et al. 2019; C. Zhang et al. 2021]. In transcript alignment [K. P. Sankar et al. 2006; Cour et al. 2008; P. Sankar et al. 2009; Y. Zhu et al. 2015; Makarand Tapaswi et al. 2015b], where instead of an action list, scripts describing a series of events in the video are given, the goal is to assign each of the script texts to the appropriate segment (shot) of the video. More closely related to our goal are methods that seek a global alignment between sequences with soft Dynamic Time Warping (DTW) [Cuturi and Blondel 2017]. The recent Drop-DTW [Dvornik et al. 2021] proposes to handle outliers in the sequences by allowing the alignment process to automatically skip certain steps. This is similar to our aim of identifiying non-alignable sentences. However, since in HowTo100M the order of the alignable sentences does not follow the original order of the subtitles, this rules out the use of DTW-type approaches.

Co-training and Self-training are common techniques for unsupervised and weakly supervised learning. Co-training [Blum and Mitchell 1998] builds two models to learn the different views of the data, while using one to expand the training set for the other. It has recently been used for representation learning [Yonglong Tian et al. 2020a; Han et al. 2020b]. Self-training refers to the process of training on pseudo-labels generated from a model's own predictions. It has been used for image classification [Caron et al. 2018; Caron et al. 2020; Yuki M Asano et al. 2020], object detection [Cinbis et al. 2015], and machine translation [K. He et al. 2020]. Our work is related to this line of research, where the TAN and the auxiliary network self-correct the noisy annotations, such that both networks can gradually improve their performance by training on cleaner data.

Supervised Action Segmentation & Detection have been extensively studied on numerous video datasets, *e.g.* Breakfast-Action [Hilde Kuehne et al. 2014], YouCook2 [L. Zhou et al. 2018a], Charades [Sigurdsson et al. 2016], ActivityNet [Heil-

bron et al. 2015], EPIC-Kitchens [Damen et al. 2020]. For segmentation, the goal is to densely classify each time point of the video into one of the pre-defined action categories [Hilde Kuehne et al. 2014; M. Rohrbach et al. 2012; Fathi and Rehg 2013; Bhattacharya et al. 2014; Bojanowski et al. 2014; Singh et al. 2016; Lea et al. 2017; P. Lei and Todorovic 2018; Farha and Gall 2019; M.-H. Chen et al. 2020]. Research has focused on designing effective modules to capture dependencies between different video chunks [Singh et al. 2016; Lea et al. 2017; P. Lei and Todorovic 2018; Farha and Gall 2019]. For detection, the goal is to localize the sparsely distributed action segments, *i.e.* annotation is non-contiguous. In general, there are two-stage approaches that consist of a separate action proposal stage and a classification stage [Shou et al. 2016; Huijuan Xu et al. 2017; Y. Zhao et al. 2017; Chao et al. 2018; T. Lin et al. 2018], and one-stage approaches that combine both [Yeung et al. 2016; Nawhal and G. Mori 2021].

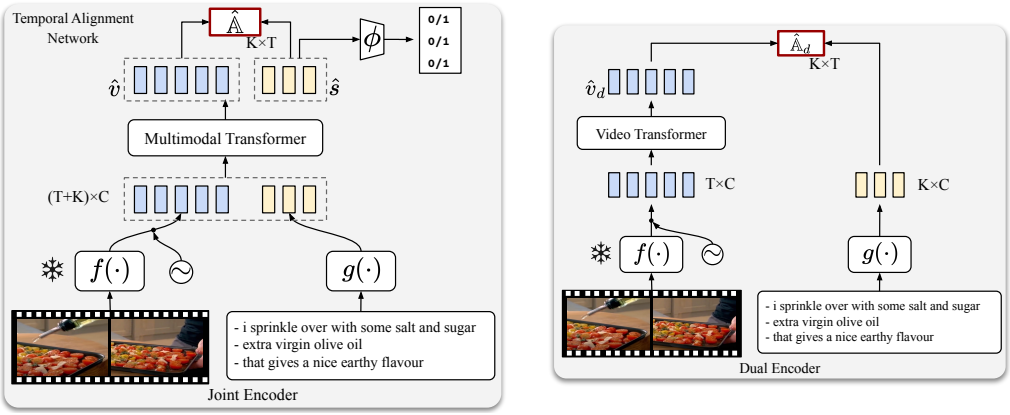


Figure 7.2: *Left*: The Temporal Alignment Network (TAN) takes an untrimmed long video as input, and first extracts the visual and textual features by a pre-trained 3D ConvNet ($f(.)$) and a pre-trained text module ($g(.)$). The visual features and textual features are concatenated and passed into a Multimodal Transformer Encoder, a.k.a. *joint encoder*, where the attention can capture the interaction between the visual and textual modalities. A linear head ϕ classifies the alignability of the output text embedding. *Right*: To train the TAN on noisy instructional videos, we build an auxiliary *dual encoder*, which takes the same visual and textual features as input, but only use a Video Transformer Encoder to process the video data with self-attention. For both TAN and the dual encoder, similarity matrices $\hat{\mathbb{A}}, \hat{\mathbb{A}}_d$ are computed between the output text features and the output visual features respectively, which are used at the co-training stage, as introduced in Section 7.3.4.

7.3 Method

We start by describing the problem scenario in Sec 7.3.1, followed by the architecture for our proposed alignment network in Sec 7.3.2. In Sec 7.3.3, we describe a naïve training procedure on raw instructional video, with the text-video correspondence provided by YouTube ASR, despite the considerable noise. In Sec 7.3.4, we

present the co-training method, that exploits the *mutual agreement* between the alignment network and an auxiliary dual encoder, and is able to simultaneously denoise and learn from the noisy narrated instructional videos.

7.3.1 Problem Scenario

Given an untrimmed instructional video $\mathcal{X} = \{\mathcal{I}, \mathcal{S}\}$, where $\mathcal{I} = \{I_1, I_2, \ldots, I_T\}$ refers to the corresponding video with T frames, and $\mathcal{S} = \{S_1, \ldots, S_K\}$ denotes the K given sentences (ordered by time). For each sentence, we also have their timestamps obtained from YouTube ASR (*e.g.* $[t_k^{\text{start}}, t_k^{\text{end}}]$ for the k-th sentence). In this paper, our goal is to train a temporal alignment network on a video dataset of instructional videos, which takes the videos and sentences as inputs, and outputs a textual-visual similarity matrix ($\hat{\mathbb{A}}$), as well as an alignability score for each sentence:

$$\{\hat{y}, \hat{\mathbb{A}}\} = \Phi(\mathcal{X}; \Theta), \quad \hat{\mathbb{A}} \in \mathbb{R}^{K \times T} \tag{7.1}$$

where $\hat{y} \in \mathbb{R}^{K \times 2}$ refers to binary scores for all sentences, indicating whether the sentence is alignable. $\hat{\mathbb{A}}$ denotes the similarity matrix between frames and the given sentences, where for any alignable sentence it should emit a higher score with its corresponding video timestamps than others, and Θ are the parameters of the model.

7.3.2 Temporal Alignment Network (TAN)

As shown in Figure 7.2 (left), the alignment network takes a video sequence and its associated narration / text sentences as input, and attends to the long temporal contexts in both, in order to: (i) determine if a sentence is alignable with the video ($\hat{y}$), and (ii) output the alignment matrix ($\hat{\mathbb{A}}$). Next, we describe the alignment network, consisting of a visual-textual backbone, Multimodal Transformer, and alignability prediction module.

Visual-Textual Backbone. Given a long instructional video (*e.g.* 64s) with its associated sentences, we first extract the visual and textual features with pre-trained networks. Specifically, based on MIL-NCE [Miech et al. 2020], we use their pre-trained S3D-G backbone to extract video features, and a 2-layer MLP with

the word2vec embeddings [Mikolov et al. 2013a] to extract sentence features.

$$v = f(\mathcal{I}) \in \mathbb{R}^{T \times C} \qquad s = g(\mathcal{S}) \in \mathbb{R}^{K \times C} \tag{7.2}$$

v, s refer to the computed video and text features respectively, and each is of dimension C, in general, $T \gg K$.

Multimodal Transformer. This module jointly processes the visual-textual features (v, s) with a multi-layer Transformer Encoder, which iteratively attends to both modalities to establish the text-to-video correspondence:

$$[\hat{v}; \hat{s}] = \Phi_{\text{MT}}([v + \text{TE}; s]) \tag{7.3}$$

where Φ_{MT} refers to the Multimodal Transformer Encoder, TE denotes the learnable temporal embedding, $\hat{v} \in \mathbb{R}^{T \times C}$ and $\hat{s} \in \mathbb{R}^{K \times C}$ are the output visual and textual embeddings from the Multimodal Transformer, and the "[;]" symbol denotes concatenation. The alignment matrix $\hat{\mathbb{A}} \in \mathbb{R}^{K \times T}$ is computed via cosine similarity:

$$\hat{\mathbb{A}}_{[i,j]} = \frac{\hat{s}_i \cdot \hat{v}_j}{\|\hat{s}_i\| \|\hat{v}_j\|} \tag{7.4}$$

Alignability Prediction Module. Apart from estimating the alignment matrix, another main functionality of the alignment network is to infer whether a particular sentence is alignable or not. This is achieved by training a single linear layer $(\phi(\cdot))$ on the textual features, as shown in Figure 7.2 (left):

$$\hat{y} = \phi_{\text{align}}(\hat{s}) \tag{7.5}$$

where $\hat{y} \in \mathbb{R}^{K \times 2}$ refers to the binary predictions for all sentences, deciding if the sentence is alignable or not.

7.3.3 Training

In this section, we describe a naïve training procedure for the alignment network with contrastive learning, on the instructional videos with YouTube ASR timestamps. Note that, at this stage, all the sentences have their corresponding video timestamps, and are treated as alignable. Hence, the alignability prediction mod-

ule can not be trained here.

Temporal Correspondence. For a video with K sentences, we directly convert its YouTube ASR results into 1D binary masks, with 1's at the timestamps where the sentence is being spoken by the demonstrator, *i.e.*, $\mathcal{Y} = \{m_1, \ldots, m_K\}$, where $m_i \in \mathbb{R}^{1\times T}$. The objective is therefore to jointly optimize the visual-textual embedding, such that the similarity score between the sentence and its corresponding visual frames is maximised. The training objective is constructed as:

$$\mathcal{L}_{\text{TC}} = -\sum_{k=1}^{K} \log \frac{\sum_{i\in\mathcal{P}_k} \exp\left(\hat{\mathbb{A}}_{[k,i]}/\tau\right)}{\sum_{i\in\mathcal{P}_k} \exp\left(\hat{\mathbb{A}}_{[k,i]}/\tau\right) + \sum_{j\in\mathcal{N}_k} \exp\left(\hat{\mathbb{A}}_{[k,j]}/\tau\right)} \tag{7.6}$$

where $\mathcal{P}_k \in \{m_k = 1\}$, $\mathcal{N}_k \in \{m_k = 0\}$ refer to the sets consisting of positive and negative pairs, respectively. $\mathcal{L}_{\text{TC}}$ resembles a variant of the InfoNCE loss [van den Oord et al. 2018].

Discussion. Given the groundtruth annotation for alignment, optimizing $\mathcal{L}_{\text{TC}}$ would be trivial. However, on raw instructional videos where the provided YouTube ASR timestamps are highly unreliable with an extremely high noise ratio, naïvely optimising $\mathcal{L}_{\text{TC}}$ leads to sub-optimal results, as will be demonstrated in Section 7.5.2.

In general, the noise sources from the raw instructional videos can be mainly categorised into three types, as shown in Figure 7.1: *First*, the majority of the given sentences are actually not correlated to the video content (unalignable), *e.g.* greeting, chatting; *Second*, there is an alignment offset, in that the temporal interval of the spoken sentence rarely aligns with the video segments it refers to; *Third*, the demonstrator often makes statements that do not follow the same order as their action, which rules out the use of DTW-type approaches.

7.3.4 Co-Training

In this section, we propose a novel *co-training* method to both denoise the instructional videos and train the alignment network. Specifically, we introduce a dual encoder (Section 7.3.4), which can be seen as a collaborator to the alignment network. This procedure is detailed below.

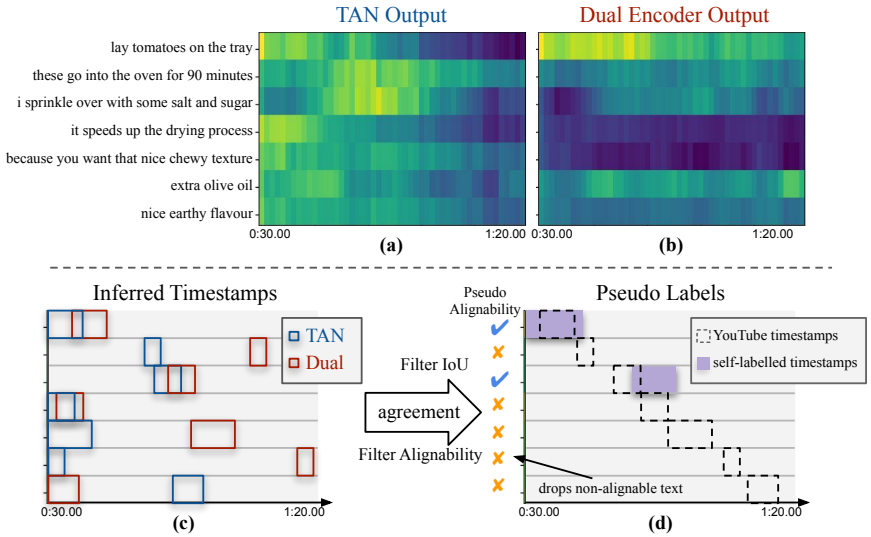


Figure 7.3: Illustration of denoising by mutual agreement. The video sample is the same as in Figure 7.1. **(a)**: The alignment matrix $\mathbb{A}$ from the TAN after Stage-1 training. **(b)**: The alignment matrix $\mathbb{A}_d$ from the dual encoder after Stage-1 training. **(c)**: The most alignable timestamps are inferred from both alignment matrices. **(d)**: By filtering the IoU of pseudo-timestamps and filtering the alignable/non-alignable text denoted by ✓ / ✗, the model dynamically chooses aligned temporal segments to train, and ignores non-alignable ones. For this example, the self-labelling process corrects the timestamps of the 1st and 3rd sentences, and marks the rest of the sentences as non-alignable. It roughly matches the human judgement of the alignment as shown in Figure 7.1. The alignment matrix values shown here are computed from the trained model-A in Table 7.1.

Dual Encoder

As shown in Figure 7.2 (right), the dual encoder independently processes the visual features with a Transformer Encoder [Vaswani et al. 2017]. It is designed to be complementary to the alignment network: for example, the dual encoder is fast and lightweight, which enables training on a large number of visual-text pairs, however, it only allows both modalities to communicate at the end, hence is unable to capture the textual contexts, and it is more sensitive to detect unaligned texts; while the proposed TAN, consisting of a Multimodal Transformer, always has access to both modalities, and can learn to establish visual-textual correspondence within the network. Despite being beneficial for the temporal alignment task, the TAN is slow and computationally demanding, limiting its ability for contrasting with large-scale and diverse negative visual-textual pairs. Formally, for the dual encoder, we have:

$$\hat{v}^d = \Phi_{\text{D}}(v + \text{TE}) \in \mathbb{R}^{T \times C} \tag{7.7}$$

where Φ_{D} refers to the Video Transformer, and TE is the learnable temporal embedding for supplying the temporal ordering. A textual-visual cosine similarity

matrix $\hat{\mathbb{A}}_d \in \mathbb{R}^{K \times T}$ from the dual encoder is computed as:

$$\hat{\mathbb{A}}_{d[i,j]} = \frac{s_i \cdot \hat{v}_j^d}{\|s_i\| \|\hat{v}_j^d\|} \tag{7.8}$$

Denoising by Mutual Agreement

To denoise the YouTube ASR annotations, we generate pseudo-labels (both the alignability and the timestamps) by verifying the *mutual agreement* between the output alignment matrices from the alignment network and dual encoder, *i.e.* $\hat{\mathbb{A}}$ and $\hat{\mathbb{A}}_d$. The verification process is executed in three steps:

(a) Infer Timestamps. During training, for each sentence, we use the two output alignment matrices $\hat{\mathbb{A}}, \hat{\mathbb{A}}_d \in \mathbb{R}^{K \times T}$ (Figure 7.3-a,b) to infer the most plausible aligned timestamps. To avoid outlier points, for the k-th sentence, we scan its corresponding similarity row by averaging the scores within a temporal window, this window is of the same size as its original YouTube timestamp label, *i.e* sentence by the demonstrator.

Afterwards, we pick the most confident prediction by taking the argmax. Note that, such operation ends up with a single temporal window with the same duration as the YouTube timestamp. That is to say, we only shift the temporal position of the original YouTube label to its most confident prediction. At this step, for the k-th sentence, we obtain two 'shifted' timestamps $\hat{m}_k$ and $\hat{m}_k^d$, one from the alignment network, the other from the dual encoder respectively, as shown in Figure 7.3-c.

(b) Alignment overlap using IoU. Given the inferred timestamps for sentence k, we compute an Intersection-over-Union (IoU) score to measure the agreement between the shifted timestamps:

$$\text{IoU-score}_k = \frac{\hat{m}_k \cap \hat{m}_k^d}{\hat{m}_k \cup \hat{m}_k^d} \tag{7.9}$$

A high IoU score indicates the sentence is very likely to be aligned with the inferred timestamps. For a batch, we filter the sentences with a positive IoU score, and update their timestamps by the union of their inferred timestamps $\hat{m}_k \cup \hat{m}_k^d$. Empirically, we find this operation roughly updates the timestamps for about 30% of the sentences. For the sentences with zero IoU score, we keep their YouTube times-

tamps unchanged. Such an operation ends up with a set of updated timestamps $\{\hat{m}'_1, \ldots, \hat{m}'_K\}$ for all sentences.

In addition, to reflect each sentence's alignability, we can compute an average cosine similarity score falling into the new temporal segment. Formally, for the k-th sentence,

$$\epsilon_k = \frac{1}{\sum \hat{m}'_k} \sum \hat{m}'_k \cdot (\hat{\mathbb{A}} + \hat{\mathbb{A}}_d)_{[k,:]} \tag{7.10}$$

where ϵ_k refers to the alignment score. To put it simply, if a sentence has positive IoU-score, we compute its align-score within the union of inferred timestamps; if it has zero IoU-score, we compute its align-score within its original YouTube timestamps.

(c) Filter Alignability. To filter the alignability scores, *i.e.*, $\{\epsilon_1, \ldots, \epsilon_K\}$, we introduce a hyper-parameter $\alpha \in [0, 1]$, within a sample batch, we treat the sentences with the top $100\alpha\%$ of align-score as positive, and the bottom $100(1-\alpha)\%$ sentences as negative. This gives binary pseudo-labels for alignability, denoted as y_{pseudo}. The alignability prediction module can thus be trained for binary classification with a cross-entropy loss (as shown in Figure 7.2), *i.e.*, $\hat{\mathcal{L}}_{\text{Alignability}} = \text{CE}(\hat{y}, y_{\text{pseudo}})$.

Intuitively, this is to say, a sentence is treated as being alignable if both the alignment network and the dual encoder agree the sentence has a high similarity with its corresponding time stamps. Also the $\mathcal{L}_{\text{TC}}$ (Equation 7.6) is only trained for the top $100\alpha\%$ of the sentences. In our experiments, we sweep $\alpha \in \{0.25, 0.5, 0.75\}$.

Training Cycle

To summarize, the training can be divided into two stages. At the first stage (S1: Initialization), both the alignment network and dual encoder are trained with $\mathcal{L}_{\text{TC}}$ using the given YouTube timestamps as labels. Once warmed up, the new pseudo-labels will be generated from the mutual agreement between alignment network and dual encoder *on the fly*, and starts the second stage training (S2: Co-Training), with $\mathcal{L}_{\text{total}} = \hat{\mathcal{L}}_{\text{TC}} + \hat{\mathcal{L}}_{\text{Alignability}}$. Note that it is not necessary to iterate S1 and S2, because in S2 the quality of pseudo-labels can be improved along the training with an EMA mechanism (introduced next). By default in our experiments, we train S1 for 50k iterations, and train S2 for another 50k iterations. It accounts for

8 epochs on HTM-370K per stage.

Self-labelling with EMA

Naïvely using the *mutual agreement* between the alignment network and dual encoder for co-training can lead to trivial solutions, where the alignment network and the dual encoder learn to "collaborate" with each other and assign high similarity scores to certain fixed timestamps. We avoid this 'collapse problem' by keeping an Exponential Moving Average (EMA) of the model similar to BYOL [Grill et al. 2020]. The EMA branch is only slowly updated and used to generate the agreements for denoising as introduced in Section 7.3.4. The main branch is trained with the updated timestamps and alignability. We use the same momentum coefficient as that from BYOL in our experiments (0.99). By default, all evaluations use the main branch.

7.4 Experiments

In this paper, we train the proposed temporal alignment network on a subset of the HowTo100M dataset [Miech et al. 2019]. To start, we first describe the data preparation process, and present the annotated visual-textual aligned subset of HowTo100M (named **HTM-Align**) for evaluation. Then we describe the implementation details and ablation studies for the alignment task.

7.4.1 Data Preparation

HowTo100M is a large-scale instructional video dataset crawled from YouTube, consisting of around 1.2M videos and their generated text transcripts from speech (ASR). The start-end timestamps of each sentence are provided by ASR, but they are often *not semantically aligned* with the visual scene (Figure 7.1).

HTM-370K (Training)

We mostly use a subset of the original HowTo100M for training, with 370K videos from the 'Food & Entertaining' categories, consisting of 32% of the videos of the entire HowTo100M dataset. Apart from the mis-alignment issue, we also find three other issues in the subtitles: incorrect language translation, duplicated text, and incomplete sentence fractions. As dataset pre-processing, we conduct an automatic curation with open-sourced BERT-based model. The full details of automatic

curation can be found in the Appendix.

After automatically processing and filtering out low-quality subtitles, we end up with a subset of 370K instructional videos, thus the name **HTM-370K**. Note that all the cleaning steps are automatic, using models trained with self-supervised learning. We attribute the pre-processing of HowTo100M as a small contribution, and we will make all cleaned video IDs and subtitles publicly available.

HTM-Align (Evaluation)

We randomly pick 80 videos from the HTM-370K as a hold-out testing set for evaluation purpose. These videos range from 2 to 16 minutes, totalling 10 hours. We manually label the alignability for each sentence, *i.e.* binary annotation. For those alignable ones, we further align them to the video segments with start-end timestamps. In total, 49K sentences are manually examined, with 13K of them being manually aligned. On average each video contains 61 sentences, and 17 of them are visually aligned.

Unlike the existing YouCook2 benchmark, where annotators only rephrase fixed recipe steps as the action description, **HTM-Align** includes random instructional videos without a fixed recipe, and are adopted from the demonstrators' narration with minor modification, hence containing large diversity on both videos and texts. The details of the annotation and examples can be found in the Appendix.

7.4.2 Implementation Details

During training, we adopt a pre-trained S3D (released by [Miech et al. 2020]) as the video encoder. Specifically, the S3D network outputs a single feature vector (1024D) for every 16 frames, when the videos are decoded with 16fps, this accounts for 1 feature per second without temporal overlap. For the text encoder, we use Bag-of-word (BoW) based on Word2Vec embeddings. By default, in each video we randomly sample a temporal window of 64 seconds (which is 64 continuous visual features, we also tried 32s and 128s in ablation study), and the corresponding subtitles within this window. We train the model with AdamW optimizer and 10^{-4} learning rate, with a batch size of 64 videos. Full implementation details are in the Appendix.

Basic Setting				Training Stages		Stage2 Settings	Aligned-HTM	
model	dataset	length (#sec)	#tfm layers	S1:Init	S2:Self	Threshold α	R@1↑	ROC-AUC↑
CLIP	YFCC-400M	–	–	–	–	–	16.8	71.7*
MIL-NCE	HTM-Full (uncurated)	–	–	–	–	–	31.3	73.1*
A	HTM-370K	64	6-6	✓	✗	–	45.8	73.0*
B	HTM-370K	64	3-3	✓	✗	–	42.3	72.6*
C	HTM-370K	64	6-6	✓	✓	0.25	42.5	79.7
D	HTM-370K	64	6-6	✓	✓	0.5	**49.4**	82.4
E	HTM-370K	64	6-6	✓	✓	0.75	48.8	82.2
F	HTM-370K	32	6-6	✓	✓	0.5	41.1	77.5
G	HTM-370K	128	6-6	✓	✓	0.5	48.4	81.8
H	HTM-Full	64	6-6	✓	✓	0.5	49.2	**82.6**

Table 7.1: **Alignment results on the HTM-Align dataset.** *: since the model does not have a binary classifier for alignability, for each sentence, we take its maximum logits over time as the alignability measurement to compute ROC-AUC. For the '# tfm layers' column, we show the number of transformer encoder layers we use for the TAN and the dual encoder. CLIP denotes the ViT-B/32 model from [Radford et al. 2021] and MIL-NCE denotes the S3D model from [Miech et al. 2020].

7.5 Alignment Results

In this section, we report the experimental results for our proposed temporal visual-textual alignment task on the **HTM-Align** dataset. In detail, during inference, given the video with a sequence of sentences by the demonstrator, we take the alignment matrix from our alignment network, $\hat{\mathbb{A}} \in \mathbb{R}^{K \times T}$, with K, T indicating the number of sentences and video timestamps respectively.

7.5.1 Metrics

We measure two metrics for the alignment task, Recall@1 and ROC-AUC value. The **Recall@1** metric is a 'pointing game' as introduced in [Zhukov et al. 2019]. Specifically, for a considered sentence, if its maximally matched video timestamp falls into the groundtruth segment, it is counted as being successfully recalled. The recall scores are then averaged across all the text segments. The alignability prediction is a binary classification problem as introduced in section 7.3.1, we use ROC curve and report the **Area-Under-the-Curve** value (ROC-AUC).

7.5.2 Ablation Study

In this section, we investigate the effects of multiple design choices and discuss the results.

Comparing with baseline. In Table 7.1, the first two rows are the baselines from CLIP (ViT-B/32) [Radford et al. 2021] and MIL-NCE [Miech et al. 2020].

Specifically, we compute the alignment similarity matrix using their textual and visual encoders, normalize the score following their pretrained paradigms, and compute the R@1 on top of the alignment matrix. Note that for ROC-AUC, since CLIP and MIL-NCE do not have a specific binary classifier, for each text, we directly use its maximum similarity score (across the time axis) as an indicator of alignability. First, CLIP [Radford et al. 2021] is performing significantly worse than others on this alignment task. A possible reason is that CLIP has only been trained on images, thus lacks video dynamics. MIL-NCE is a strong baseline which has short-term temporal modelling (up to 3.2s) and was trained end-to-end on HowTo100M. In our model-A, we take the pre-extracted visual and textual feature from MIL-NCE, and train the transformers on the HTM-370K dataset to learn a longer temporal context (*e.g.* 64s) for the alignment task. Our result (model-A 45.8 R@1 vs MIL-NCE 31.3 R@1) shows that longer temporal context is useful for this alignment task.

Effect of Transformer Depth. For both the alignment network and dual encoder, we use 6-layer transformers by default, as a balance between performance and training cost. In model-B we also tried using 3-layer transformer and found it performs worse than 6-layer transformer (model-B vs A). Using more than 6 layers takes more memory and sacrifices batch size.

Effect of Co-Training. In the model-{D,E}, we apply the Stage-2 training (co-training) based on the model-A. We observe that co-training brings a clear performance gain for the alignment task (model-{D,E} vs. model-A, 3-4% boost on R@1), confirming the effectiveness of the denoising procedure. Note that model-C does not perform well due to the choice of alignability threshold α, explained next.

Effect of Alignability Thresholds. For the choice of alignability threshold α (as introduced in Section 7.3.4), which reflects a balance of data noise and diversity in the co-training procedure, our model-{C,D,E} show $\alpha = 0.5$ and $\alpha = 0.75$ work similarly well for alignment metrics and $\alpha = 0.5$ is slightly better for the R@1 metric. However $\alpha = 0.25$ leads to much worse performance. We conjecture that a low value of α limits the diversity while training $\mathcal{L}_{\text{TC}}$ (*i.e.* $\mathcal{L}_{\text{TC}}$ learns from only 25% of the sentences).

Effect of Training Data. In model-H, we train the co-training stage on the au-

tomatically curated HTM-Full dataset, which includes all other non-cooking categories from HowTo100M comparing with HTM-370K. Comparing model-H with D on the alignment task, adding out-of-domain videos does not harm the alignment performance on our curated subset.

Effect of Input Video Length. In Table 7.1, we vary the length of the input video to show if our alignment network benefits from the longer video context. Indeed, the alignment network gets better performance when increasing the input video length from 32 to 64 seconds (model-D vs model-F). We conjecture that sampling longer input video introduces more *alignable* sentences, helps to reduce the temporal ambiguity for other sentences. However, further increasing the input video length to 128 seconds gives a similar alignment performance (model-G vs model-D), we conjecture this is due the reduced batch size in training, and the far-away visual context (*i.e.* 2 minutes or further) is less relevant for aligning the sentence.

Method	Trained on BF	F-Acc↑	IoU↑	IoD↑
MIL-NCE [Miech et al. 2020]	✗ (ZS)	59.3	46.8	65.1
Ours (S1+S2)	✗ (ZS)	**65.1**	**50.6**	**68.6**
D^3TW [C.-Y. Chang et al. 2019]	✓	57.0	-	56.3
CDFL [Jun Li et al. 2019]	✓	63.0	45.8	63.9
DP-DTW [X. Chang et al. 2021]	✓	67.7	50.8	66.5
Ours (S1+S2)	✓	**68.3**	**51.7**	**69.3**

Table 7.2: **Temporal alignment on the Breakfast-Action (BF) dataset.** We split the previous methods into two groups. For the upper group, the model has not seen any samples in Breakfast-Action dataset since Breakfast-Action videos are not downloadable from YouTube. For the lower group, the model is trained with weak supervision on the Breakfast-Action training set.

Method	Trained on YC2	R@1↑	R@5↑	R@10↑	Median R↓
ActBERT [L. Zhu and Y. Yang 2020a]	✗ (ZS)	9.6	26.7	38.0	19
MIL-NCE [Miech et al. 2020]	✗ (ZS)	15.1	38.0	51.2	10
MIL-NCE [Miech et al. 2020]†	✗ (ZS)	13.9	36.3	48.9	11
TaCo [J. Yang et al. 2021]	✗ (ZS)	19.9	43.2	55.7	8.0
Ours (S1)	✗ (ZS)	16.8	41.3	54.8	8.0
Ours (S1 + S2)	✗ (ZS)	**20.1**	**45.5**	**59.5**	**7.0**

Table 7.3: **Text-based video retrieval on the YouCook2 (YC2) dataset.** ZS refers to "zero-shot", where the alignment network is only trained on HTM-180K, and directly evaluated on YouCook2. †: reproduced in [L. Zhou et al. 2021]. For our results, **S1** denotes only training Stage-1 (initialization), which is the model-A from Table 7.1. **S1+S2** denotes training with two stages (initialization followed by co-training), which is the model-E from Table 7.1.

7.6 Downstream Tasks

Apart from evaluating the alignment task on **HTM-Align**, we also test our alignment network on other downstream tasks. Specifically, we evaluate on temporal action alignment (using the alignment network) and text-based video retrieval (using the dual encoder due to speed considerations [Miech et al. 2021]). We also evaluate linear action classification on the backbone feature to show the effect of auto-aligned dataset. See the Appendix for full details.

Datasets. To evaluate the alignment network, we use *Breakfast-Action* [Hilde Kuehne et al. 2014] and *YouCook2* [L. Zhou et al. 2018a] for downstream tasks. To evaluate the end-to-end representation learning, we use *UCF101* [Soomro et al. 2012], *HMDB51* [Hildegard Kuehne et al. 2011] and *K400* [Kay et al. 2017].

Temporal Alignment on Breakfast-Action. Given a video with multiple actions and the corresponding action descriptions, the model needs to densely label each video timestamp with one given text description, often defined as weakly-supervised action segmentation by the community. Following previous work [Ding and C. Xu 2018; C.-Y. Chang et al. 2019; Jun Li et al. 2019; X. Chang et al. 2021], we report three metrics: frame-wise accuracy (**F-Acc**), segment-wise Intersection-over-Union (**IoU**) and Intersection-over-Detection (**IoD**). Please refer to Appendix for more details.

We evaluate our method with both the zero-shot and finetune settings. In the former case, our alignment network was *only* trained on HTM-370K, and directly evaluated on Breakfast; while in the latter, we finetune our alignment network on Breakfast with a soft-DTW loss [Cuturi and Blondel 2017] stacked on top of the output alignment matrix for 50 epochs. During inference, the alignment network takes as input a single video and the given list of action labels, i.e. crack egg', 'fry egg', *etc*, and outputs the alignment matrix $\mathbb{A}$, which is passed through a DTW, ending up with the action boundaries.

As shown in Table 7.2, in the zero-shot setting, our proposed alignment network surpasses the strong baseline (MIL-NCE) by a large margin on all metrics ($> 3\%$), and even achieves comparable results to those supervised approaches. After finetuning, we see a further performance boost, obtaining state-of-the-art results.

Text-based Video Retrieval on YouCook2. We evaluate the model for text-

based video retrieval on the YouCook2 dataset. For this task, we pass each pre-cropped video segment through the *dual encoder*, and take the visual features (v_{enc}) from the Video Transformer Encoder. Also we pass the task description phrases into the dual encoder and take the word2vec features. For each query text, we rank the video segments based on cosine similarity among 3.5k candidates. Following previous works [Miech et al. 2019; Miech et al. 2020], we report retrieval Recall @{1,5,10} and Median Rank.

As shown in Table 7.3, under the zero-shot setting, where the proposed alignment network was only trained on HTM-370K, our model surpasses previous works by a clear margin, especially on R@5, R@10 and Median R. Importantly, the results show that the co-training stage substantially improves the performance of the *dual encoder* (R@1 20.1 vs 16.8), also our method surpasses the baseline method MIL-NCE by a large margin (R@1 20.1 vs 15.1).

End-to-end Representation Learning. The output of the Temporal Alignment Network can be used to clean-up (automatically align) long-video datasets. We use model-H to automatically align the HTM dataset, and finetune the S3D-word2vec backbone **end-to-end** with an Info-NCE loss on the auto-aligned text-video pairs for only 10 epochs. We evaluate the visual representation by linear probing on action classification, and find the auto-aligned HTM timestamps benefits the end-to-end video representation. We refer the readers for more details in the Appendix.

Settings	Backbone	UCF101	HMDB51	K400
reported by [Miech et al. 2020]	S3D	82.7	53.1	-
reproduce of [Miech et al. 2020]	S3D	82.1	55.2	55.7
finetuned with TAN	S3D	**83.2**	**56.7**	**56.2**

Table 7.4: **Linear-probing action classification performance.** We evaluate the end-to-end trained visual representations on UCF101, HMDB51 and K400 by linear probing (LP). We show the reported LP results from [Miech et al. 2020] (1st row), our reproduction of LP results of the official S3D weights (2nd row), and our finetuned S3D performance with auto-aligned HTM under the exact same setting (3rd row).

7.7 Conclusion

In summary, we introduce a temporal alignment network, with a co-training method for denoising the instructional video datasets. To evaluate the alignment accuracy we introduce a new benchmark dataset with 10 hours of videos, with the narrations manually aligned to corresponding video timestamp. When evaluat-

ing on **HTM-Align**, Breakfast-Action, YouCook2, under zero-shot or finetune settings, our model achieves state of the art results, surpassing multiple strong baselines (MIL-NCE, CLIP). We also show the proposed method can clean-up (by improving the alignment) large-scale public datasets and further improve the visual-textual backbone representations.

Acknowledgement

This research is funded by EPSRC Programme Grant VisualAI EP/T028572/1, a Royal Society Research Professorship RP\R1\191132, and a Google-DeepMind Scholarship. We thank Charig Yang, Guanqi Zhan and Chuhan Zhang for proof-reading.

Appendices

Appendices for this chapter can be found in the online version of the paper. [1]

Statement of authorship

A statement of authorship for this paper is provided in Appendix A.

[1] https://www.robots.ox.ac.uk/~vgg/research/tan/

Chapter 8

AutoAD: Movie Description in Context

The paper is a technical report presenting ongoing works, written in 2022.

AutoAD: Movie Description in Context

Tengda Han[1] Max Bain[1] Arsha Nagrani[1,3]

Gül Varol[1,2] Weidi Xie[1,4] Andrew Zisserman[1]

[1]Visual Geometry Group, University of Oxford

[2]LIGM, École des Ponts, Univ Gustave Eiffel, CNRS

[3]Google Research [4]Shanghai Jiao Tong University

Abstract

The objective of this paper is an automatic Audio Description (AD) model that ingests movies and outputs AD in text form. Generating high-quality movie AD is challenging due to the dependency of the descriptions on context, and the limited amount of training data available. In this work, we leverage the power of pretrained foundation models, such as GPT and CLIP, and only train a mapping network that bridges the two models for visually-conditioned text generation. In order to obtain high-quality AD, we make the following four contributions: (i) we incorporate context from the movie clip, AD from previous clips, as well as the subtitles of the current shot; (ii) we address the lack of training data by pretraining on large scale datasets, where visual or contextual information are unavailable, e.g. text-only AD without movies or visual captioning datasets without context; (iii) we improve on the currently available AD datasets, by removing label noise in the MAD dataset, and adding character naming information; and (iv) we obtain strong results on the movie AD task compared with previous methods.

Context AD:
As Karen stares groomly out of the window, a man approaches toying with a lighter. She turns her head, and finds Jack standing beside her.

Subtitles:
> Can I buy you a drink?
> Yeah I'd love one. Sit down.

Target AD: He takes the seat opposite, then places his lighter on the table

Figure 8.1: **Movie audio description (AD)** consists of sentences describing movies for the visually impaired. Note how it is heavily influenced by various types of context - the visual frames, the previous AD, and the subtitles of the movie.

8.1 Introduction

That of all the arts, the most important for us is the cinema.

Vladimir Lenin

One of the long term aims of computer vision is to understand long-form feature films. There has been steady progress towards this aim with the identification of characters by their face and voice [Bojanowski et al. 2013; A. Brown et al. 2021; Everingham et al. 2006; M. Tapaswi et al. 2012; Q. Huang et al. 2018], the recognition of their actions and inter-actions [Laptev et al. 2008; Marszałek et al. 2009; Patron-Perez et al. 2010; Vondrick et al. 2016a], of their relationships [Kukleva et al. 2020], and 3D pose [Pavlakos et al. 2022]. However, this is still a long way away from story understanding. Movie *Audio Description (AD)*, the narration describing visual elements in movies, provides a means to evaluate current movie understanding capabilities. AD was developed to aid visually impaired audiences, and is typically generated by experienced annotators. The amount of AD on the internet is growing due to more societal support for visually impaired communities and its inclusion is becoming an emerging legal requirement.

AD differs from image or video captioning in several significant respects [Research and Center 2013], bringing its own challenges. First, AD provides dense descriptions of important visual elements *over time*. Second, AD is always provided on a separate sound track to the original audio track and is highly *complementary* to it. It is complementary in two ways: it does not need to provide descriptions

of events that can be understood from the sound track alone (such as dialogue and ambient sounds), and it is constrained in time to intervals that do not overlap with the dialogue. Third, unlike dense video captioning, AD aims at *storytelling*; therefore, it typically includes factors like a character's name, emotion, and action descriptions.

In this work, our objective is automatic AD generation – a model that takes continuous movie clips as input and outputs AD in text form. This is a relatively unexplored task in the vision community with previous work targeting ActivityNet videos [Y. Wang et al. 2021], a very different domain to long-term feature films with storylines, and the LSMDC challenge [A. Rohrbach et al. 2015].

As usual, one of the challenges holding back progress is the lack of suitable training data. Paired image-text or video-text data that is available at scale, such as alt-text [Radford et al. 2021; Sharma et al. 2018] or stock footage with captions [Bain et al. 2021], does not generalize well to the movie domain [Bain et al. 2022]. However, collecting high-quality data for movie understanding is also difficult. Researchers have tried to hire human annotators to describe video clips [X. Chen et al. 2015; J. Xu et al. 2016; Krishna et al. 2017b] but this does not scale well. Movie scripts, books and plots have also been used as learning signals [Bojanowski et al. 2013; Sigurdsson et al. 2016; Y. Zhu et al. 2015] but they do not ground on vision closely and are limited in number.

In this paper we address the AD and training data challenges by – Spoiler Alert – developing a model that uses temporal context together with a visually conditioned generative language model, while providing new and cleaner sources of training data. To achieve this, we leverage the strength of large-scale language models (LLMs), like GPT [Radford et al. 2019], and vision-language models, like CLIP [Radford et al. 2021], and integrate them into a video captioning pipeline that can be effectively trained with AD data.

Our contributions are the following: (i) inspired by ClipCap [Mokady et al. 2021] we propose a model that is effectively able to leverage both temporal context (from previously generated AD) and dialogue context (in particular the names of characters) to improve AD generation. This is done by bridging foundation models with lightweight adapters to integrate both types of context; (ii) we address the lack of large-scale training data for AD by pretraining components of our model on partially missing data which are typically available in large quantities e.g. text-only

AD without movie frames, or visual captioning datasets without multiple sentences as context; (iii) we propose an automatic pipeline for collecting AD narrations at scale using speaker-based separation; and finally (iv) we show promising results on automatic AD, as seen from both qualitative and quantitative evaluations – in doing so, we achieve impressive zero-shot results on the LSMDC multi-description benchmark comparable to the finetuned state-of-the-art.

8.2 Related Works

Image Captioning. Image captioning is a long-standing problem in computer vision [X. Chen and Zitnick 2014; Donahue et al. 2015; Karpathy and Fei-Fei 2015; Kiros et al. 2014; Lu et al. 2018; Anderson et al. 2018; X. Chen et al. 2015]. Early pioneering works learn to associate images and words within a limited vocabulary and a set of images [Barnard and Forsyth 2001; Barnard et al. 2003; Lavrenko et al. 2003]. Large-scale image captioning datasets have been collected by scraping images from the internet and their corresponding alt-texts with quality filters as a post-processing [Sharma et al. 2018]. In doing so, strong joint image-text representations can be learned [Radford et al. 2021], and image captioning from raw pixels, with impressive results [J. Yu et al. 2022; Junnan Li et al. 2022]. Recent work [Mokady et al. 2021; Nukrai et al. 2022] learns a bridge between strong joint image-text representations (CLIP) and the natural language representation (GPT-2) for image captioning, obtaining promising results that generalise well across domains. In this work, we extend this approach to perform automatic AD from videos.

Video Captioning. Video captioning presents additional challenges due to the lack of quality large-scale video-text data and increased complexity from the temporal axis. Early video caption datasets [D. Chen and Dolan 2011; J. Xu et al. 2016] adopt manual annotations, a far from scalable collection method. ASR (automated speech recognition) from YouTube instructional videos is collected at scale for video-language datasets [Miech et al. 2019], but contains high levels of noise due to the weak correspondence between the narration and visual content. VideoCC [Nagrani et al. 2022] transfers captions from images to videos, but this method is still limited by the existing seed image captioning dataset used. Earlier video captioning models lack generalisation capabilities due to limited training

data [Venugopalan et al. 2015; Park et al. 2019]. Some recent methods [Seo et al. 2022; G. Huang et al. 2020; H. Luo et al. 2020] train on ASR from the HowTo100M dataset, while others expand image-text representations [M. Tang et al. 2021] to multiple frames.

A task more related to AD is that of dense video captioning [Krishna et al. 2017a], which involves producing a number of captions and their corresponding grounded timestamps in the video. To enrich inter-task interactions, recent works for this task [Chadha et al. 2021; S. Chen and Y.-G. Jiang 2021; C. Deng et al. 2021; Yehao Li et al. 2018; Mun et al. 2019; Rahman et al. 2019; Shen et al. 2017; Shi et al. 2019; Jingwen Wang et al. 2018; T. Wang et al. 2021; L. Zhou et al. 2018b] jointly train both a captioning and localization module. Our task differs in that the captions are: made with the intent to aid storytelling; specific to the movie domain; and complementary to the audio track.

Visual Storytelling. Most similar in vein to the AD task is visual storytelling [T.-H. Huang et al. 2016; Junnan Li et al. 2020; Ravi et al. 2021], in which the goal is to generate coherent sentences for a sequence of video clips or images. LSMDC [A. Rohrbach et al. 2017] proposes the multi-description task of generating captions for a set of clips from a movie, with character names anonymized. In contrast, movie AD takes as input a continuous long video and describes the visual happenings complementary to the story, characters, dialogue and audio. Most similar to our model is TPAM [Y. Yu et al. 2021] which prompts a frozen GPT-2 with local visual features. Ours differs in that: (i) it is not restricted to local visual context but rather global by recurrently conditioning on previous outputs; and (ii) we additionally pretrain GPT on in-domain text-only AD data.

Movie Understanding. Previous works investigate storyline understanding by aligning movies to additional data sources such as plots [Xiong et al. 2019; Y. Sun et al. 2022], books [Makarand Tapaswi et al. 2015a; Y. Zhu et al. 2015], scripts [Papalampidi et al. 2019], and YouTube summaries [Bain et al. 2020]. However, these sources are limited in number and often do not closely relate to the visual elements in the frame. Using existing movie AD as the data source for videos is an emergent direction for movie understanding. LSMDC [A. Rohrbach et al. 2015], M-VAD dataset [Torabi et al. 2015] and MPII-MD [A. Rohrbach et al. 2015], gather AD and scripts from movies to provide captions for short video clips, several seconds in duration. QuerYD [Oncescu et al. 2021] provides high-quality

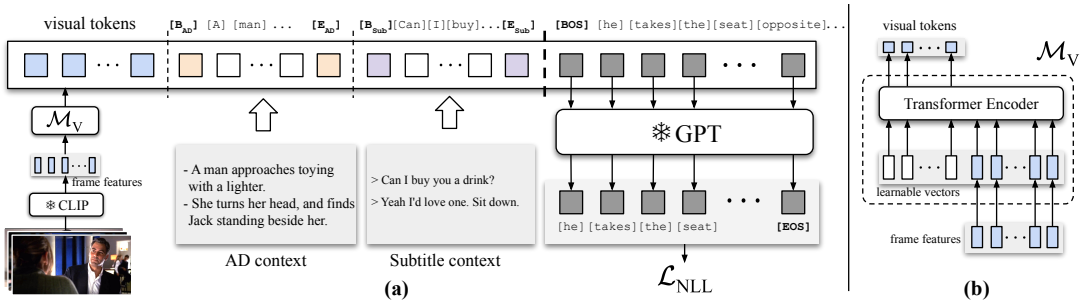


Figure 8.2: **(a) Overview of AutoAD:** AutoAD consists of a *frozen* visual encoder (CLIP) and a *frozen* LLM (GPT) for generating captions. We introduce a lightweight mapping network to map CLIP features into visual tokens, which are then combined with previous AD context and subtitle context, before being fed into the GPT model. $\mathcal{M}_V$ refers to the visual mapping network, [B$_*$] and [E$_*$] denote the learnable special tokens for contextual AD and subtitle sequences. **(b) Detail of the visual mapping network:** A transformer encoder takes as input multiple frame features and outputs a few visual tokens which are further feed to a text generation model.

textual descriptions for longer videos by scraping AD from YouDescribe [Research and Center 2013], an online community of AD contributors. Recently, the MAD dataset [Soldan et al. 2022] collects movie AD at scale to provide dense textual annotations for movies with the focus on visual grounding task.

Prompt Tuning and Adapters. Originally for language modelling, prompt tuning is a lightweight approach to adapt a pretrained model to perform a downstream task. Early works [T. Brown et al. 2020; Lester et al. 2021; X. L. Li and P. Liang 2021; Ju et al. 2022] learn prompt vectors that are shared within the targeted dataset and task. A similar line of works to ours is *visual-conditioned* prompt tuning, in which the prompt vectors are conditioned on the visual inputs. Visual-conditioned prompts are used for adapting pretrained image-language models [Bahng et al. 2022; M. Jia et al. 2022], and for few-shot learning [Tsimpoukelli et al. 2021; Alayrac et al. 2022]. Training lightweight feature adapters between pretrained vision and text encoders is another approach to adapt pretrained models [P. Gao et al. 2021; Renrui Zhang et al. 2021]. The adapter layers can also be inserted into the pretrained language model in an interleaved way [A. Yang et al. 2022]. Our work adopts prompt tuning in order to condition a language generation model on visual information (frames), and textual context (subtitles and previous AD).

8.3 Method

Given a long-form movie $\mathcal{V}$ segmented into multiple short clips $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$, our goal is to generate the audio description (AD) in text form for every movie clip. Specifically, for the i-th movie clip consisting of multiple frames $\mathbf{x}_i = \{\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_N\}$, we aim to produce text $\mathcal{T}_i$ that describes the visual elements in such a way that helps the visually impaired follow the story line. To this purpose, an ideal AD generation system must be able to exploit the full contextual information leading up to the i-th movie clip. One method for this, which we adopt, is to use previous AD $\mathcal{T}_{t<i}$ and subtitles $\mathcal{S}_{t\leq i}$ to generate the text $\mathcal{T}_i$. In the following sections, we first give an overview of our visual captioning pipeline with prompt tuning (Sec. 8.3.1), followed by our contextual components (Sec. 8.3.2), and finally the pretraining methods with partial data (Sec. 8.3.3).

8.3.1 Visual Captioning with Prompt Tuning

In order to describe our method, we first present the typical pipeline for an image captioning model, and then detail how we extend this to ingest multiple frames and additional text context. Given an image-caption pair $\{\mathcal{I}_i, \mathcal{C}_i\}$, where the caption consists of a sequence of language tokens $\mathcal{C}_i = \{c_1, c_2, ..., c_k\}$, the standard objective of an image captioning model is to generate text tokens $\hat{\mathcal{C}}_i$ that are close to the target $\mathcal{C}_i$. Technically, the captioning models are trained to maximize the joint probability of predicting the ground-truth language tokens, or equivalently minimize the following negative log-likelihood (NLL) loss,

$$\mathcal{L}_{\text{NLL}} = -\log p_\theta(\mathcal{C}_i|\mathbf{h}_{\mathcal{I}_i}) = -\log p_\theta(c_1, c_2, ..., c_k|\mathbf{h}_{\mathcal{I}_i})$$

where θ denotes the parameters of the model, and $\mathbf{h}_{\mathcal{I}_i}$ denotes the extracted image features of $\mathcal{I}_i$. Previous works like ClipCap [Mokady et al. 2021] fit a powerful text generation model and visual encoding model into this image captioning pipeline. Specifically, strong visual encoding models, such as CLIP [Radford et al. 2021], are used to extract the visual features from the input image $\mathbf{z}_i = f_{\text{CLIP}}(\mathcal{I}_i)$, then a visual mapping network $\mathcal{M}_{\text{V}}$ is trained to map the visual features to 'prompt vectors' that adapt to the text generation model, $\mathbf{h}_{\mathcal{I}_i} = \mathcal{M}_{\text{V}}(\mathbf{z}_i)$. Finally these prompt vectors $\mathbf{h}_{\mathcal{I}_i}$ are fed to a pretrained text generation model, such as GPT [Radford et

al. 2019], for the captioning task. We adapt this visual captioning pipeline, which uses pretrained feature extractor CLIP and langauge model GPT, for movie AD generation and propose key components that support contextual understanding.

8.3.2 Benefiting from Temporal Context

Here, we describe how we extend this single-frame captioning model to include different forms of context, including multiple frames, previous AD text, and subtitles. Compared to image captioning where the annotation describes 'what is in the image', movie AD describes the visual happenings in the scene that are relevant to the broader story – often centered around events, characters and the interactions between them. Factors like these cannot be accurately described from a static image alone and therefore a successful automatic AD system must utilize the context of prior events and character interactions.

To tackle these temporal dependencies, we propose to include three components to incorporate the essential contextual information from movies: (i) immediate visual context in the current movie shot (multiple frames), (ii) the previous movie AD, and (iii) the movie subtitles. The architecture of our model is shown in Fig. 8.2.

Multiple frames (immediate visual context). In contrast to the image captioning method, the visual mapping network $\mathcal{M}_{\mathrm{V}}$ takes as input multiple frame features from the current movie shot $\mathbf{x}_i$ rather than a single image feature, and outputs prompt vectors for the movie shot,

$$\mathbf{h}_{\mathbf{x}_i} = \mathcal{M}_{\mathrm{V}}(\{\mathbf{z}_1, ..., \mathbf{z}_N\}); \quad \mathbf{z}_i = f_{\mathrm{CLIP}}(\mathcal{I}_i).$$

In detail, the mapping network consists of a multi-layer transformer encoder that enables modelling temporal relations among multiple frame features, as shown in Fig 8.2.

Previous AD text. The sequence of events leading up to the present contain contextual information which are crucial for generating AD of current scene that helps the viewer follow the story. We input this contextual knowledge to our model in the form of the past ADs. Specifically, our model takes the past K movie ADs $\{\mathcal{T}_{i-K}, ..., \mathcal{T}_{i-1}\}$ to generate the AD for the current shot. The past movie ADs are a few sentences, which are first concatenated into a single paragraph, then tokenized

and converted to a sequence of word embeddings. Inspired by the design of special tokens in language models, we wrap the context AD embeddings with *learnable* special tokens to indicate the beginning and end of the AD sequence. Formally, the contextual AD embedding is a sequence,

$$\mathbf{h}_{\text{AD}} = [\texttt{B}_{\text{AD}}; \mathbf{h}_{\mathcal{T}_{i-K}}; ...; \mathbf{h}_{\mathcal{T}_{i-1}}; \texttt{E}_{\text{AD}}], \tag{8.1}$$

where $\texttt{B}_{\text{AD}}$ and $\texttt{E}_{\text{AD}}$ are the learnable special tokens indicating the beginning and end, the symbol ';' denotes concatenation, and $\mathbf{h}_{\mathcal{T}_j} \in \mathbb{R}^{n \times C}$ denotes the word embedding of the j-th movie ADs.

Previous subtitles. Our model also takes the movie subtitles as additional contextual information, which can be sourced either from the official movie metadata or automatically transcribed with an ASR model. The character dialogues, contained with the subtitles, provide complementary information to movie description, including the character names, relationships and emotions. Similar to the context ADs, we concatenate multiple subtitle sentences into a single paragraph and wrap them with learnable special tokens. Practically, since the timing of movie AD does not overlap with the subtitles, we take the most recent L subtitles within a certain time range as the context,

$$\mathbf{h}_{\text{Sub}} = [\texttt{B}_{\text{Sub}}; \mathbf{h}_{\mathcal{S}_{i-L}}; ...; \mathbf{h}_{\mathcal{S}_{i-1}}; \texttt{E}_{\text{Sub}}].$$

Due to the weak correlation between the subtitles and the visual elements in the scene, we also experiment with a variant that only encodes the character names occurring in the recent subtitles.

Summary. Overall, the movie AD for the current shot $\mathcal{T}_{\mathbf{x}_i}$ is generated by conditioning on all the previously described visual and contextual information using a pretrained GPT. The conditional information is fed to GPT as prompt vectors as shown in Fig. 8.2. The model is trained with NLL loss,

$$\mathcal{L}_{\text{NLL}} = -\log p_\Theta(\mathcal{T}_{\mathbf{x}_i} | \mathbf{h}_{\mathbf{x}_i}, \mathbf{h}_{\text{AD}}, \mathbf{h}_{\text{Sub}}). \tag{8.2}$$

During training, we input the ground-truth past AD. During inference, we experiment with two methods to incorporate the past AD: an **oracle** setting where

Manual Verification*	She stands and the little warrior takes in her size, about twice his own.	As she steps past him, he defensively grips his spear	Leia sits on a moss covered log.
MAD-v1[Soldan et al.]	Angola, she stands in the Little Warrior, takes in her size about twice his own.	As she steps past me. Defensively grips his spear.	I'm not gon na. Leah sits on a Moss covered log.
MAD-v2 (ours)	She stands and the little warrior takes in her size about twice his own.	As she steps past him, he defensively grips his spear.	Leia sits on a moss-covered log.

Figure 8.3: **Qualitative comparison of MAD annotations.** We compare the original MAD-v1 [Soldan et al. 2022] and our proposed MAD-v2. Note MAD-v1's erroneous transcriptions of AD and dialogue leakage (highlighted in red text). The samples are taken from Star Wars VI: Return of the Jedi (1983) [Marquand 1983]. *We verify this example by manually transcribing the AD narration from the audio track.

the *ground-truth* past ADs are used in Eq. 8.2 to generate the current AD, and a **recurrent** setting where the *predicted* past ADs are used instead.

8.3.3 Pretraining with Partial Data

A major challenge for generating AD is the lack of training data, since the model requires the corresponding visual, textual and contextual data to all be jointly trained. However since our model is modular, components of it can be pretrained with *partial data* – when a certain type of data is missing, the remaining modules can still be trained. We experiment with partial-data pretraining under two settings: visual-only pretraining and AD-only pretraining.

Visual-only Pretraining. In the absence of contextual data, the visual mapping network can be pretrained with the abundant image captioning or (short) video captioning datasets. In this case, the context modules (both contextual AD and subtitles) are deactivated. The training objective of Eq. 8.2 is turned into $\mathcal{L} = -\log p_\Theta(\mathcal{T}_{\mathbf{x}_i}|\mathbf{h}_{\mathbf{x}_i})$ for visual-only pretraining. Note that the langauge model is kept frozen here since we find image/video captioning datasets have a clear domain gap with movie AD in both the vision and text modalities.

AD-only Pretraining. Movie AD datasets with corresponding visual information (e.g. frames or frame features) are limited at scale due to potential copyright issues. However, abundant *text-only* movie ADs are available online as described in Sec. 8.5. In the absence of visual data, the contextual AD module and the language model can still be pretrained. The training objective in this case becomes $\mathcal{L} = -\log p_\Theta(\mathcal{T}_{\mathbf{x}_i}|\mathbf{h}_{\text{AD}})$, which is similar to training a story completion objective [Mostafazadeh et al. 2016] by finetuning GPT on *text-only* movie AD data

but with a few additional special tokens. This text-only movie AD pretraining is also related to [Gururangan et al. 2020], which shows a second stage of language model pretraining on in-domain data improves downstream performance.

8.4 Denoising MAD Dataset

Our main objective is to generate movie audio descriptions. For this goal, the model is trained on the MAD training set [Soldan et al. 2022], a dataset of AD caption-video clip pairs from 488 movies. MAD provides the video data in the form of CLIP visual features in order to avoid copyright restrictions. The AD annotations for each movie are automatically collected from AudioVault[1], a large open-source database of audio files containing the full-length original movie track mixed with the AD narrator's voice. The MAD authors transcribe this data using ASR, and also have access to the official DVD subtitles. Their automated method then uses *text-based* speaker separation of the transcribed audio by using subtitles to know when dialogue is present, and assuming all other speech is AD.

This however introduces *significant noise* because: (i) they use an outdated ASR model resulting in erroneous transcriptions; and (ii) official DVD subtitles are not exhaustive of all speech in the movie and thus such a method frequently misidentifies character dialogue as AD narration, thereby introducing substantial noise to the annotations (an example is provided in Fig. 8.3). Further, obtaining official subtitles from DVDs presents additional challenges when collecting this data at scale.

We propose an improved automated data collection method for AD, requiring only the audio track as input (no DVD subtitles), that tackles both of these issues by using *audio-based* speaker separation and an improved ASR model. We then use this method to collect improved annotations for the MAD dataset. Briefly, taking the mixed audio containing both AD narrations and original movie sound track as input, our automated AD collection pipeline contains four stages: (1) speaker diarization [Bredin et al. 2020; Bredin and Laurent 2021] to segment the temporal boundaries of speech and assign speaker IDs to each segment via clustering; (2) speech-to-text with OpenAI's whisper [Radford et al. 2022] – an advanced ASR model; (3) labelling the speaker ID of the AD narrator by selecting the cluster with

[1] https://audiovault.net

Dataset	Total movies	Total duration (hrs)	Total AD captions	Subtitles	Visual Features
QueryD [Oncescu et al. 2021]	-	207	31K	✗	✓
LSMDC [A. Rohrbach et al. 2017]	200	147	128K	✗	✓
MAD-v1 [Soldan et al. 2022]	488	892	280K	✓	✓
MAD-v2 (ours)	488	892	264K	✓	✓
AudioVault (ours)	7,057	12,510	3.3M	✓	✗

Table 8.1: **Statistics of Audio Description datasets.** We report relevant statistics to compare our MAD-v2 and Audiovault datasets.

the lowest proportion of first-person pronouns (e.g. 'I' and 'we'); and finally (4) synchronization of the segment timestamps with the visual features by comparing audio. Further details are in the Appendix.

Henceforth we refer to the original MAD annotation [Soldan et al. 2022] as **MAD-v1** and our new denoised annotations as **MAD-v2**. A qualitative comparison is shown in Fig. 8.5, we find that our MAD-v2 is much more robust and contains less errors and less character dialogue leakage. Both LSMDC and MAD-v1 post-process their annotations by replacing character names in the annotations with 'someone' via entity recognition, and release both variants of annotations which we refer to as **`Named`** and **`Unnamed`**. Similarly, we propose two variants of our denoised annotations:

MAD-v2-`Named`: This version contains the raw collected AD narrations *without* any post-processing on the character names.

MAD-v2-`Unnamed`: Following the character name anonymisation performed in early works, we identify character names using a Named Entity Recognition (NER) model [Polle n.d.] and replace them with 'someone'.

8.5 Partial Pretraining with AudioVault Dataset

Paired AD and corresponding visual data are difficult to obtain especially due to movie copyrights, whereas a large number of movie ADs audio tracks are available online for free (e.g. AudioVault). To demonstrate the effect of partial pretraining in Sec. 8.3.3, we collect a large-scale *text-only* movie AD dataset from AudioVault. In detail, we source mixed audio files from over 7,000 movies from AudioVault that are not included in MAD-v1, and use a similar denoising pipeline described in Sec. 8.4 to obtain the movie ADs (detailed in Appendix). Additionally we obtain a proxy for the movie subtitles by assuming the ASR from all the non-AD

speakers are the characters' dialogues. To ensure no test-time leakage, we remove all movies present in either LSMDC or MAD from the dataset.

Overall, our AudioVault dataset is an order of magnitude greater than prior AD datasets (see Table 8.1), from which we provide two sets of data:

AudioVault-AD. The AD narrations from AudioVault and their corresponding timestamps within each movie, totalling 3.3 million AD utterances.

AudioVault-Sub. The subtitles data from AudioVault and their corresponding timestamps within each movie, totalling 8.7 million subtitle utterances.

Both our MAD-v2 and AudioVault datasets will be released publicy.

8.6 Experiments

In this section we first outline the experimental details for the AD task, the datasets used for training & testing, the architectural details, and the evaluation metrics (Sec. 8.6.1). We then report results and discuss the findings on the aforementioned evaluations, perform ablations on our model, and compare to prior works (Sec. 8.6.2).

8.6.1 Implementation Details

Dataset

Training Dataset. **CC3M** (Conceptual Caption) [Sharma et al. 2018] is a large image alt-text dataset that contains 3.3M web images. **WebVid** [Bain et al. 2021] is a large video-caption dataset that contains 2.5M short stock footage videos. We use them for the partial-data pretraining for visual modules. Additionally, we use our **AudioVault-AD** to pretrain the textual modules, as described in Sec. 8.3.3. For the main Movie AD task, we train with original **MAD-v1** and our cleaned version **MAD-v2**, detailed in Sec. 8.4.

Test Dataset. **LSMDC** [A. Rohrbach et al. 2015] contains 118K short video clips with descriptions from 202 movies, in which 182 of them are public. The original MAD-val&test split inherits LSMDC annotations after filtering out 20 lower-quality movies, resulting in 162 movies from all the LSMDC train/val/test splits. However, training models with LSMDC train/val is a standard practice used by other works. In order to facilitate future evaluation on movie AD task, we propose an evaluation split that excludes LSMDC train/val movies from these 162

movies, which gives an evaluation set containing 13 movies, refered as **MAD-eval** henceforth. Similarly, we use both **MAD-eval-`Named`** and **MAD-eval-`Unnamed`** versions. The 'Unnamed' version corresponds to standard LSMDC annotation style – where the characters' titles and names in the descriptions are replaced by the word 'someone'; the 'Named' version is constructed from the original character names provided by LSMDC. Additionally, subtitles are not provided with MAD val/test or LSMDC, so we transcribe them from the full length audio tracks using whisper.

Architecture

For **visual features**, we use the CLIP ViT-B-32 model [Radford et al. 2021], which is a 12-layer transformer encoder that outputs 1×512 feature vectors for each input frames. These features are provided by the MAD dataset. For **visual mapping network**, we use a 2-layer transformer encoder with 8 attention heads and a 512 hidden dimensions, followed by a linear projection layer that projects 512-d features into 768-d. We use ten prompt vectors. For **language model**, we use GPT-2 [Radford et al. 2019], specifically the version from HuggingFace. The GPT-2 model takes as input 768-d token embeddings, passes through a 12-layer transformer encoder with a causal attention map, and outputs the next token embedding for every input token. We limit the generated number of tokens at 36, since most movie ADs are less than 36 tokens. The GPT-2 is frozen in most of our experiments unless otherwise stated. Each special token (e.g. $\texttt{B}_{\text{AD}}$) is a learnable 768-d vector. We take at most 64 past AD tokens and 32 subtitle tokens, and short text samples are padded. Specifically for subtitles, we take the most recent four dialogues within a one minute time window.

Training and Inference Details

On the MAD-v1 and MAD-v2 datasets, we use a batch size of 8 sequences, each of which contains 16 consecutive video-AD pairs from a movie. Overall that gives us 8×16 video-AD pairs for every batch. From each video clip, 8 frame features are uniformly sampled. By default, the model is trained for 10 epochs. One epoch means the model has seen *all* the audio descriptions once. Additional implementation details are in the Appendix.

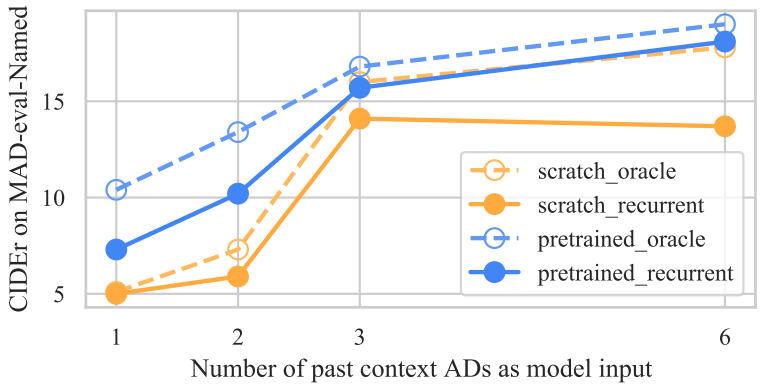


Figure 8.4: **Effect of the length of context AD.** We use the model 'V+AD' in Table 8.2, and train with different number of past AD sentences. 'scratch' indicates no partial-data pretraining; 'pretrained' refers to pretraining with text-only AudioVault-AD.

We use the AdamW optimizer [Ilya Loshchilov 2019] and a cosine-decay learning rate schedule with a linear warm-up. The starting learning rate is 10^{-4} and is decayed to 0. For each experiment, we use a single Nvidia A-40 for training. For text generation, greedy search and beam search are commonly used sampling methods. We stop the text generation as long as a full stop mark is predicted, otherwise we limit the sequence length to 67 tokens. We use beam search with a beam size of 5 and mainly report results by the top-1 beam-searched outputs, since beam search performs slightly better than greedy search on multiple scenarios. Note that under the 'recurrent' setting, we feed the past greedy-searched text outputs to the model to generate the current AD, which we find gives more stable results.

Evaluation Metrics

To evaluate the quality of text generation compared to the ground-truth, we use classic metrics including ROUGE-L [C.-Y. Lin 2004] (**R-L**), CIDEr [Vedantam et al. 2015] (**C**) and SPICE [Anderson et al. 2016] (**S**). We also report BertScore [T. Zhang et al. 2020] (**BertS**), which evaluates word matching between candidate sentence and reference sentence with pretrained BERT embeddings. A higher value indicates a better text generation compared with the ground-truth.

Temporal Context	Partial Data Pretrain	R-L	C	S	BertS
None (1 frame)	None	7.1	4.0	1.0	13.2
V (8 frames)	None	9.3	6.7	2.4	15.6
	CC3M [Sharma et al. 2018]	9.9	8.4	2.4	16.8
	WV [Bain et al. 2021]	9.9	10.0	2.0	17.3
V+AD	None	11.8 (13.3)	13.7 (17.8)	5.6 (5.8)	18.8 (22.1)
	AV-AD	13.0 (13.9)	18.1 (19.0)	4.7 (4.8)	22.9 (23.7)
	WV, AV-AD	12.7 (13.9)	18.5 (21.9)	4.3 (4.8)	24.2 (23.8)
V+AD+Sub	WV, AV-AD	12.9	16.1	4.9	22.2
V+AD+SubN*	WV, AV-AD	13.4	17.0	5.3	23.6

Table 8.2: **Ablative experiments of our AD captioning method.** We ablate our model with different types of temporal context and partial pretraining. All models are trained on MAD-v2-`named` and evaluated on MAD-eval-`named`. For models with AD context we report recurrent results with oracle in parentheses. 'V' refers to visual context by taking multi-frame inputs, 'WV' refers to WebVid2M dataset, 'AV-AD' here refers to our partial-data pretraining with text-only AudioVault-AD dataset. *'SubN' denotes the variant of subtitle module that only takes names as input.

MAD Train Set		MAD-eval-`Unnamed`				MAD-eval-`Named`			
		R-L	C	S	BertS	R-L	C	S	BertS
v1	`Unnamed`	15.1	12.7	9.5	22.4	12.7	15.9	4.7	22.0
	`Named`	11.3	10.9	3.0	24.0	12.8	17.0	5.2	21.8
v2	`Unnamed`	**15.9**	**14.5**	**10.5**	**26.7**	12.9	18.0	4.7	22.0
	`Named`	11.4	10.0	3.1	22.5	**13.3**	**17.8**	**5.8**	**22.1**

Table 8.3: **Effect of denoising MAD training data annotation.** We train a model with 6 contextual ADs on MAD-v1 [Soldan et al. 2022] or MAD-v2 sources without any pretraining. The model is evaluated on both the **`named`** and **`unnamed`** versions of MAD-eval under the **oracle** setting. Cross-domain testing results (when the model is trained and tested on different types of annotations) are provided for reference and marked in gray.

Methods	Pretraining Data	R-L	C	S	BertS
ClipCap [Mokady et al. 2021]	CC3M	8.5	4.4	1.1	11.8
CapDec* [Nukrai et al. 2022]	AV-AD	8.2	6.7	1.4	14.3
AutoAD (ours)	AV-AD	**13.0**	18.1	**4.7**	22.9
AutoAD (ours)	AV-AD & WebVid	12.7	**18.5**	4.3	**24.2**

Table 8.4: Compared with other works on movie AD generation task on MAD-v2. We obtain results from other methods by finetuning their models on MAD-v2-`Named` dataset, and evaluated on MAD-eval-`Named`. *CapDec [Nukrai et al. 2022] proposes text-only pretraining to adapt the style for text generation, we pretrained their model on the text-only AudioVault-AD dataset then applied it to MAD-v2.

Methods	Paired Training Data	C	M
Baseline [Park et al. 2019]	LSMDC	11.9	8.3
TAPM [Y. Yu et al. 2021]	LSMDC	15.4	**8.4**
AutoAD (ours)	MAD-v2-`Unnamed`	16.7	7.4
AutoAD (ours)	MAD-v2-`Unnamed` & LSMDC	**17.5**	7.5

Table 8.5: **Results on the LSMDC 2019 Multi-Sentence Description public test set.** We report our method with different amounts of training data and without subtitles for comparison under similar settings. Official challenge metrics (CIDEr and METEOR) are reported with the 'sentence' setting as described in [A. Rohrbach et al. 2015; Y. Yu et al. 2021].

Figure 8.5: **Qualitative examples of automatically generated AD by AutoAD.** We highlight AD predictions under both the oracle and recurrent settings. Previous AD context is shown in gray. For ease of visualisation, a single frame from each movie clip is shown with subtitles overlaid. Samples are taken from Titanic (1997) [Cameron 1997].

8.6.2 Experiments on Movie Audio Descriptions

Effect of Temporal Context. In Table 8.2 we show that visual context from the nearby frames brings a clear gain for the AD task (C 6.7 vs 4.0). AD context provides a consistent performance improvement under both oracle (C 17.8 vs 6.7) and recurrent settings (C 13.7 vs 6.7). Note that we find feeding AD context as text tokens works better than training a textual feature mapping network, we conjecture the ADs in their original text form carry the most key information like the names and places. However, subtitle context provides no gain for our model (C 16.1 vs 18.1) under the recurrent setting, which we attribute to the very weak correspondence between the visual elements in the scene and the character dialogue. When the subtitles are filtered and contain only character names (denoted as 'SubN'), they provide a slight performance gain (R-L 13.4 vs 13.0, S 5.3 vs 4.7). Since the subtitles used are without speaker identities, the model may struggle to know which character in the frame spoke each subtitle. Overcoming these challenges will be considered in future work.

Effect of MAD data cleaning. Table 8.3 demonstrates the benefit of our proposed MAV v2 annotations over v1, confirming the qualitative findings. Training the AD model with context on v2 outperforms training on v1 under all settings (both named and unnamed) by a significant margin. Since the v2 annotations are fewer in number than MAD-v1, this suggests they are indeed less noisy and result in AD captioning models with improved performance.

Effect of pretraining with partial data. In Table 8.2, we find that **visual-only pretraining** on open-domain vision-text data provides clear gains (CIDEr 8.4 vs 6.7 for CC3M, and 10.0 vs 6.7 for WebVid). But considering the size of visual samples, the improvement is not data-efficient. We attribute this to the large domain gap between movie AD and classical visual caption annotations like CC3M or WebVid2M. The **text-only pretraining** of our model improves performance significantly. For the recurrent AD context model, AudioVault-AD pretraining increases CIDEr by a relative 32% (from 13.7 to 18.1), reaching comparable performance to oracle AD context *without* pretraining. These results indicate the greater importance of adapting to the text style and context over adapting the visual features. The combination of the visual module after visual-only pretraining (WebVid) and the textual modules after text-only pretraining (AV-AD) gives further performance gain (C 21.9 vs 19.0 for the oracle setting and 18.5 vs 18.1 for recurrent).

Length of Context. In Figure 8.4 we show the effect of varying the amount of context AD that is given to the model. Longer AD context improves performance almost consistently across all settings. Note that we experiment at most 6 contextual AD sentences, which is equivalent to about 70 word embeddings in Eq. 8.1. The trend suggests longer contextual AD would further improve performance but brings extra computational cost due to the quadratic complexity of the attention operation in GPT-2.

Qualitative Results

Fig. 8.5 shows qualitative examples of our model. Under the oracle setting, the model can use the character identities easily from the past ground-truth AD

(e.g. master-at-arms), whereas under the recurrent setting, the model can only learn names from the noisy subtitles, therefore it currently mostly predicts pronouns (e.g. "he", "they") but still gets the actions ("looks") or objects ("necklace") correct.

8.6.3 Comparison with Other Works

In Table 8.4, we compare our method with previous visual captioning methods. Note that since the MAD dataset only releases the CLIP visual features, rather than the movie frames, our comparison is limited to methods that build on frozen CLIP features. We show a clear performance improvement compared to ClipCap [Mokady et al. 2021] and CapDec [Nukrai et al. 2022], for the latter the language model is also adapted to the movie AD domain by text-only pretraining. The results highlight the importance of context for movie AD.

In Table 8.5, we adapt our method to the Multi-Sentence Description task on LSMDC, in which the model takes five consecutive clips and generates five corresponding descriptions. Since the task is performed on the *unnamed* annotations, we finetune our best model in Table 8.4 with varying 0-4 context ADs as input on MAD-v2-`Unnamed` dataset and test with the *recurrent* setting. To make minimal changes, our model still takes a single clip feature at each step, whereas previous method takes all five clips together for movie description. Despite this disadvantage, we obtain competitive results on this task even without using the *manually-cleaned* LSMDC training set (C 16.7 vs 15.3), effectively *zero-shot*. The performance of the model can be further improved by additionally training on LSMDC data.

8.7 Conclusion and Future Work

This paper focuses on the automatic generation of movie AD and has made significant progress. We propose our AutoAD pipeline that incorporates contextual information. Additionally, we demonstrate the effectiveness of partial-data pretraining, a technique that could be widely applicable when full data is difficult to obtain. Further, we clean up the previous MAD dataset and collect a new text-only movie AD dataset as a pretraining resource. We note that automatic movie

AD is challenging and still far from being solved. Future work could incorporate global context of the full movie as well as utilizing the complementary information between AD and the movie audio track.

Statement of authorship

A statement of authorship for this paper is provided in Appendix A.

Chapter 9

Discussion

In this chapter, we first summarise the achievements and the impact of the presented works in this thesis (Section 9.1), and then briefly suggest directions for future works (Section 9.2).

9.1 Achievement and Impact

Dense Predictive Coding. In Chapter 2, we introduce the paper "Video Representation Learning by Dense Predictive Coding" (DPC), which is one of the pioneering works that apply contrastive loss on large-scale video representation learning. In the paper, the contrastive loss is applied on a large number of spatio-temporal features across a batch of videos, and we find a large number of features is beneficial for contrastive learning since it supplies informative hard negative samples for the learning process. Later, the benefit of large-batch contrastive learning was confirmed and scaled up on image tasks in the MoCo [K. He et al. 2020] and SimCLR [T. Chen et al. 2020] with industry-scale resources. Our early investigation of hard negative samples in contrastive learning influenced later works that pursue efficient contrastive learning signals, including our CoCLR paper [Han et al. 2020b] and other works [Yonglong Tian et al. 2020b; Afouras et al. 2020; Jabri et al. 2020]. The task of predicting future videos inspired other works that search for learning signals from the time axis of videos [Benaim et al. 2020; Jiangliu Wang et al. 2020] and use the video predictive task to detect unintentional actions [Epstein et al. 2020].

In Chapter 3, we introduce the work "Memory-augmented Dense Predictive Coding for Video Representation Learning" (MemDPC), which investigates the

multiple hypotheses problem of the future prediction task. A subsequent work [Surís et al. 2021] tackles future uncertainty by predicting future features in the hyperbolic space. Note that our public codebase is used to develop their models. Additionally, we propose to use a non-linear classification head to evaluate the feature quality rather than a commonly-used linear classifier, which was recently popularised by MAE paper [K. He et al. 2022].

The paper DPC (Chapter 2) and MemDPC (Chapter 3) are jointly cited more than 400 times. They often appear in the comparison tables of video SSL methods, serving as standard baselines of contrastive learning in video domains. The open-sourced GitHub codebase also gets more than 400 stars, providing a reliable reference for future research.

Self-supervised Co-Training. In Chapter 4, we introduce the work "Self-supervised Co-Training for Video Representation Learning" (CoCLR), which proposes to use complementary modalities to teach each other for hard sample mining under a contrastive loss. Although our CoCLR experiments with RGB and optical flow modalities, our method is universal to be applied to any complementary modalities. Indeed, CoCLR influences subsequent works that learn from visual and audio [Morgado et al. 2021; Patrick et al. 2020], visual and text [Ge et al. 2022; Qian et al. 2022b], or more than two modalities [Gurram et al. n.d.]. In [Surís et al. 2022], the visual similarity connects the text embeddings from different languages that achieves a language translation. Some works also explore complementary information within the visual modality, e.g. learning from short and wide temporal contexts [Recasens et al. 2021] and nearest neighbour within the visual modality [Dwibedi et al. 2021].

Additionally, we propose supervised contrastive learning – termed "UberNCE" in the paper, which injects the ground-truth label information in the contrastive loss, as an upper-bound oracle setting for self-supervised experiments. The concurrent paper [Khosla et al. 2020] popularised the concept and set a high baseline in the supervised image classification benchmarks.

In terms of performance, CoCLR is the first method that achieves more than 90% action classification accuracy on UCF101 dataset with visual-only pre-training. Our RGB-only action retrieval performance on UCF101 and HMDB51 remains to be state-of-the-art for almost two years – which is a big achievement in this fast-

progressing field – until the recent work [Qian et al. 2022a] that decouples static and dynamic video features with several regularisation losses. More importantly, while counterparts conduct industry-scale training with a few orders of magnitude of more computational resources, CoCLR achieves competitive performance on video benchmarks with at most 4 GPUs, which highlights its effectiveness in self-supervised training. Overall, the CoCLR paper is cited more than 200 times and the GitHub codebase has received more than 260 stars.

Efficient Training. In Chapter 5, we present the paper "Prompting Visual-Language Models for Efficient Video Understanding". In this work, we show that the prompt tuning method can be applied to image-based visual-language models like CLIP [Radford et al. 2021] to steer it for video understanding tasks. We extensively conduct experiments on 10 public benchmarks including action recognition, action localisation and text-video retrieval, across closed-set, few-shot and zero-shot scenarios, offering a reliable baseline for future research in this area. The subsequent works [M. Jia et al. 2022] extend the learnable prompts into the visual networks, and [Bahng et al. 2022] design the learnable visual prompts in the pixel space.

The work "Turbo Training with Token Dropout" in Chapter 6 shows the recent Masked AutoEncoder (MAE) is not only an efficient visual representation learning method [K. He et al. 2022; Feichtenhofer et al. 2022; Tong et al. 2022], but also a universal acceleration method for Vision Transformer architectures on video-language pre-training, finetuning for action classification, and long video classification. There are a few concurrent works that also point out the acceleration of MAE but are limited to finetuning for classification [Yue Liu et al. 2022; F. Liang et al. 2022]. The public codebase will encourage more researchers to adapt this method to accelerate their training tasks.

Temporal Alignment Networks. In terms of visual-language training, natural videos like the ones on YouTube have lots of noise and result in a low signal-to-noise ratio (SNR). Previous work points out that these challenging noises from texts and visual contents hinder the effectiveness of large-scale video training [Miech et al. 2020].

In this work, we increase the SNR of natural YouTube videos to make them

great again for visual-language training. In detail, we propose a temporal alignment network that learns to automatically remove the sentences irrelevant to the visual scene, as well as align the timestamps of relevant sentences with their corresponding visual content. As a result, we obtain the automatically aligned version of HowTo100M dataset, named 'HTM-AA'. In our paper, we show that HTM-AA benefits the backbone visual representation quality comparing with the original HowTo100M. We believe many more subsequent works on short-term or long-term video tasks would benefit from the large-scale pre-training on the HTM-AA. We have released the HTM-AA dataset to the public.

Additionally, when working on natural long-term videos, we find lots of issues with the ASR-based subtitles. For example, some English subtitles are transcribed from another non-English speech directly, resulting in meaningless phonemes in text, and the subtitles are often truncated into text segments consisting of only a few words, failed to maintain the meaning of instruction sentences. We propose to use pre-trained BERT-based language models to pre-process the subtitles, and obtain a version of HowTo100M with each subtitle in a full sentence. We have also released the text processing pipeline and the cleaned version of subtitles for future research.

Despite just being published recently, we believe the paper together with the public datasets and toolbox will inspire many subsequent works on natural long-video training.

9.2 Future Works

We conclude the thesis by outlining a few possible research directions for future works.

Long-video Understanding. In Part III we present two works on long-video tasks including instructional videos and featured movies, but long-video understanding is still a scarcely explored research area. Architectural designs and task definitions are promising research directions in this area. First, long-video architecture requires an effective memory mechanism, because loading an entire long video for processing is neither efficient for hardware nor generalisable to variable video durations. An ideal long-video memory should store multiple hierarchies of

semantic information, e.g. the action of lifting a fork can also mean eating dessert or attending a formal dinner, depending on the level of hierarchy. Additionally, a long-video memory mechanism should also store multiple modalities like vision, audio and speeches. To achieve these functions effectively, novel architectural designs are required. Second, defining tasks that require long-video understanding is difficult. Many video tasks like classical action classification and localisation are not ideal evaluations since a few frames suffice to achieve the tasks. The suitable long-video evaluation tasks should assess object awareness in space and time, as well as higher-level abstractions like reasoning actions or understanding storylines. However, how we define the annotation instructions and how we scale up long-video annotations remain to be challenging problems.

Active Environment. In Chapter 2 and 3, we describe predictive coding being applied to videos for representation learning. However, if the model learns by passively watching videos, the interaction with the environment is a missing component that limits the representation quality. Self-supervised video representation can be learned in an active environment, where the model can take action and observe the consequences by processing the video inputs. Furthermore, compared with natural web videos, the data from real-world active environments are limited in size and expensive to collect, therefore challenging for self-supervised training. As such, data generation methods that build on top of high-quality simulator engines like Kubric [Greff et al. 2022] can be developed to produce realistic visual inputs within interactive environments. Following this line, domain transfer methods can be researched in order to adapt the simulation-pretrained video representations to real-world tasks.

9.3 Conclusion

In this thesis, we develop methods to learn video representation without manual annotations. These works in conjunction with the upcoming technical progress on computations and networks, will enable us to create machines that could learn the human intelligence from the enormous amount of videos.

References

Triantafyllos Afouras, Andrew Owens, Joon Son Chung, and Andrew Zisserman (2020). "Self-supervised learning of audio-visual objects from video". In: *Proc. ECCV*.

Pulkit Agrawal, João Carreira, and Jitendra Malik (2015). "Learning to see by moving". In: *Proc. ICCV*.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Roman Reynolds Malcolm Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Marianne Samangooei Sina Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan (2022). "Flamingo: a Visual Language Model for Few-Shot Learning". In: *NeurIPS*.

Humam Alwassel, Dhruv Mahajan, Lorenzo Torresani, Bernard Ghanem, and Du Tran (2019). "Self-Supervised Learning by Cross-Modal Audio-Video Clustering". In: *arXiv preprint arXiv:1911.12667*.

Humam Alwassel, Dhruv Mahajan, Lorenzo Torresani, Bernard Ghanem, and Du Tran (2020). "Self-Supervised Learning by Cross-Modal Audio-Video Clustering". In: *NeurIPS*.

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R. Devon Hjelm (2019). "Unsupervised State Representation Learning in Atari". In: *NeurIPS*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould (2016). "Spice: Semantic propositional image caption evaluation". In: *Proc. ECCV*.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang (2018). "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering". In: *Proc. CVPR*.

Relja Arandjelović, Petr Gronát, Akihiko Torii, Tomás Pajdla, and Josef Sivic (2016). "NetVLAD: CNN architecture for weakly supervised place recognition". In: *Proc. CVPR*.

Relja Arandjelović and Andrew Zisserman (2017). "Look, Listen and Learn". In: *Proc. ICCV*.

Relja Arandjelović and Andrew Zisserman (2018). "Objects that Sound". In: *Proc. ECCV*.

Yuki M Asano, Christian Rupprecht, and Andrea Vedaldi (2020). "Self-labelling via Simultaneous Clustering and Representation Learning". In: *International Conference on Learning Representations*.

Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi (2020). "Self-labelling via simultaneous clustering and representation learning". In: *Proc. ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural machine translation by jointly learning to align and translate". In: *Proc. ICLR*.

Hyojin Bahng, Ali Jahanian, Swami Sankaranarayanan, and Phillip Isola (2022). "Exploring Visual Prompts for Adapting Large-Scale Models". In: *arXiv preprint arXiv:2203.17274*.

Max Bain, Arsha Nagrani, Andrew Brown, and Andrew Zisserman (2020). "Condensed Movies: Story Based Retrieval with Contextual Embeddings". In: *Proc. ACCV*.

Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman (2021). "Frozen in Time: A Joint Video and Image Encoder for End-to-End Retrieval". In: *Proc. ICCV*.

Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman (2022). "A CLIP-Hitchhiker's Guide to Long Video Retrieval". In: *arXiv preprint arXiv:2205.08508*.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei (2022). "BEiT: BERT Pre-Training of Image Transformers". In: *Proc. ICLR*.

Kobus Barnard, Pinar Duygulu, David Forsyth, Nando De Freitas, David M Blei, and Michael I Jordan (Feb. 2003). "Matching Words and Pictures". In: *JMLR*.

Kobus Barnard and David Forsyth (2001). "Learning the semantics of words and pictures". In: *Proc. ICCV*. IEEE.

Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T. Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel (2020). "SpeedNet: Learning the Speediness in Videos". In: *Proc. CVPR*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin (2003). "A neural probabilistic language model". In: *JMLR*.

Gedas Bertasius, Heng Wang, and Lorenzo Torresani (2021). "Is Space-Time Attention All You Need for Video Understanding?" In: *Proc. ICML*.

Subhabrata Bhattacharya, Mahdi M. Kalayeh, Rahul Sukthankar, and Mubarak Shah (2014). "Recognition of Complex Events: Exploiting Temporal Dynamics between Underlying Concepts". In: *Proc. CVPR*.

Mina Bishay, Georgios Zoumpourlis, and Ioannis Patras (2019). "Tarn: Temporal attentive relation network for few-shot and zero-shot action recognition". In: *Proc. BMVC*.

Avrim Blum and Tom Mitchell (1998). "Combining Labeled and Unlabeled Data with Co-Training". In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*.

Piotr Bojanowski, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic (2013). "Finding Actors and Actions in Movies". In: *Proc. ICCV*.

Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic (2014). "Weakly Supervised Action Labeling in Videos Under Ordering Constraints". In: *Proc. ECCV*.

Piotr Bojanowski, Rémi Lajugie, Edouard Grave, Francis Bach, Ivan Laptev, Jean Ponce, and Cordelia Schmid (2015). "Weakly-Supervised Alignment of Video With Text". In: *Proc. ICCV*.

Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool (2016). "Dynamic filter networks". In: *NeurIPS*.

Hervé Bredin and Antoine Laurent (2021). "End-to-end speaker segmentation for overlap-aware resegmentation". In: *Proc. Interspeech 2021*.

Hervé Bredin, Ruiqing Yin, Juan Manuel Coria, Gregory Gelly, Pavel Korshunov, Marvin Lavechin, Diego Fustes, Hadrien Titeux, Wassim Bouaziz, and Marie-Philippe Gill (2020). "pyannote.audio: neural building blocks for speaker diarization". In: *Proc. ICASSP*.

Andrew Brown, Ernesto Coto, and Andrew Zisserman (2021). "Automated Video Labelling: Identifying Faces by Corroborative Evidence". In: *International Conference on Multimedia Information Processing and Retrieval*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020). "Language Models are Few-Shot Learners". In: *NeurIPS*.

Uta Büchler, Biagio Brattoli, and Björn Ommer (2018). "Improving Spatiotemporal Self-Supervision by Deep Reinforcement Learning". In: *Proc. ECCV*.

James Cameron (1997). *Titanic*. United States: Paramount Pictures.

Kaidi Cao, Jingwei Ji, Zhangjie Cao, Chien-Yi Chang, and Juan Carlos Niebles (2020). "Few-Shot Video Classification via Temporal Alignment". In: *Proc. CVPR*.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze (2018). "Deep clustering for unsupervised learning of visual features." In: *Proc. ECCV*.

Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin (2019). "Unsupervised Pre-Training of Image Features on Non-Curated Data". In: *Proc. ICCV*.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin (2020). "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments". In: *NeurIPS*.

Joao Carreira and Andrew Zisserman (2017). "Quo vadis, action recognition? a new model and the kinetics dataset". In: *Proc. CVPR*.

João Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman (2018). "A Short Note about Kinetics-600". In: *arXiv preprint arXiv:1808.01340*.

João Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman (2019). "A Short Note on the Kinetics-700 Human Action Dataset". In: *arXiv preprint arXiv:1907.06987*.

João Carreira and Andrew Zisserman (2017). "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: *Proc. CVPR*.

Aman Chadha, Gurneet Arora, and Navpreet Kaloty (2021). "iPerceive: Applying Common-Sense Reasoning to Multi-Modal Dense Video Captioning and Video Question Answering". In: *Proc. WACV*.

Chien-Yi Chang, De-An Huang, Yanan Sui, Li Fei-Fei, and Juan Carlos Niebles (2019). "D^3TW: Discriminative Differentiable Dynamic Time Warping for Weakly Supervised Action Alignment and Segmentation". In: *Proc. CVPR*.

Xiaobin Chang, Frederick Tung, and Greg Mori (2021). "Learning Discriminative Prototypes with Dynamic Time Warping". In: *Proc. CVPR*.

Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A. Ross, Jia Deng, and Rahul Sukthankar (2018). "Rethinking the Faster R-CNN Architecture for Temporal Action Localization". In: *Proc. CVPR*.

David Chen and William B Dolan (2011). "Collecting highly parallel data for paraphrase evaluation". In: *Proc. ACL: human language technologies*.

Min-Hung Chen, Baopu Li, Yingze Bao, Ghassan AlRegib, and Zsolt Kira (2020). "Action Segmentation with Joint Self-Supervised Temporal Domain Adaptation". In: *Proc. CVPR*.

Shaoxiang Chen and Yu-Gang Jiang (2021). "Towards bridging event captioner and sentence localizer for weakly supervised dense event captioning". In: *Proc. CVPR*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton (2020). "A Simple Framework for Contrastive Learning of Visual Representations". In: *arXiv preprint arXiv:2002.05709*.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He (2020). "Improved Baselines with Momentum Contrastive Learning". In: *arXiv preprint arXiv:2003.04297*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick (2015). "Microsoft coco captions: Data collection and evaluation server". In: *arXiv preprint arXiv:1504.00325*.

Xinlei Chen and C. Lawrence Zitnick (2014). "Learning a Recurrent Visual Representation for Image Caption Generation". In: *arXiv*.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Yu Cheng Zhe Gan, and Jingjing Liu (2020). "Uniter: Learning universal image-text representations". In: *Proc. ECCV*.

Sumit Chopra, Raia Hadsell, and Yann LeCun (2005). "Learning a similarity metric discriminatively, with application to face verification". In: *Proc. CVPR*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy J. Colwell, and Adrian Weller (2021). "Rethinking Attention with Performers". In: *Proc. ICLR*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555*.

Ramazan Gokberk Cinbis, Jakob J. Verbeek, and Cordelia Schmid (2015). "Weakly Supervised Object Localization with Multi-fold Multiple Instance Learning". In: *IEEE PAMI*.

T. Cour, C. Jordan, E. Miltsakaki, and B. Taskar (2008). "Movie/Script: Alignment and Parsing of Video and Text Transcription". In: *Proc. ECCV*.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio (2016). "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1". In: *arXiv preprint arXiv:1602.02830*.

Ioana Croitoru, Simion-Vlad Bogolin, Marius Leordeanu, Hailin Jin, Andrew Zisserman, Samuel Albanie, and Yang Liu (2021). "Teachtext: Crossmodal generalized distillation for text-video retrieval". In: *Proc. ICCV*.

Marco Cuturi and Mathieu Blondel (2017). "Soft-DTW: a Differentiable Loss Function for Time-Series". In: *Proc. ICML*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov (2019). "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context". In: *Proc. ACL*.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray (2020). "Rescaling Egocentric Vision". In: *arXiv preprint arXiv:2006.13256*.

Chaorui Deng, Shizhe Chen, Da Chen, Yuan He, and Qi Wu (2021). "Sketch, ground, and refine: Top-down dense video captioning". In: *Proc. CVPR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805*.

Ali Diba, Vivek Sharma, Luc Van Gool, and Rainer Stiefelhagen (2019). "DynamoNet: Dynamic Action and Motion Network". In: *Proc. ICCV*.

Li Ding and Chenliang Xu (2018). "Weakly-Supervised Action Segmentation with Iterative Soft Boundary Assignment". In: *Proc. CVPR*.

Carl Doersch, Abhinav Gupta, and Alexei A Efros (2015). "Unsupervised Visual Representation Learning by Context Prediction". In: *Proc. CVPR*.

Carl Doersch and Andrew Zisserman (2017). "Multi-task Self-Supervised Visual Learning". In: *Proc. ICCV*.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell (2015). "Long-term recurrent convolutional networks for visual recognition and description". In: *Proc. CVPR*.

Jianfeng Dong, Xirong Li, Chaoxi Xu, Shouling Ji, Yuan He, Gang Yang, and Xun Wang (2019). "Dual encoding for zero-example video retrieval". In: *Proc. CVPR*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby (2020). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *Proc. ICLR*.

Alexey Dosovitskiy and Vladlen Koltun (2017). "Learning to Act by Predicting the Future." In: *Proc. ICLR*.

Nikita Dvornik, Isma Hadji, Konstantinos G. Derpanis, Animesh Garg, and Allan D. Jepson (2021). "Drop-DTW: Aligning Common Signal Between Sequences While Dropping Outliers". In: *NeurIPS*.

Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman (2021). "With a little help from my friends: Nearest-neighbor contrastive learning of visual representations". In: *Proc. ICCV*.

Sai Kumar Dwivedi, Vikram Gupta, Rahul Mitra, Shuaib Ahmed, and Arjun Jain (2019). "ProtoGAN: Towards Few Shot Learning for Action Recognition". In: *Proc. CVPR*.

Gerald M. Edelman and Joseph A. Gally (2013). "Reentry: a key mechanism for integration of brain function". In: *Frontiers in Integrative Neuroscience*.

Dave Epstein, Boyuan Chen, and Carl Vondrick (2020). "Oops! Predicting Unintentional Action in Video". In: *Proc. CVPR*.

Mark Everingham, Josef Sivic, and Andrew Zisserman (2006). ""Hello! My name is... Buffy" – Automatic Naming of Characters in TV Video". In: *Proc. BMVC*.

Yazan Abu Farha and Juergen Gall (2019). "MS-TCN: Multi-Stage Temporal Convolutional Network for Action Segmentatio". In: *Proc. CVPR*.

Alireza Fathi and James M. Rehg (2013). "Modeling Actions through State Changes". In: *Proc. CVPR*.

Christoph Feichtenhofer (2020). "X3D: Expanding Architectures for Efficient Video Recognitionion". In: *Proc. CVPR*.

Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He (2022). "Masked Autoencoders As Spatiotemporal Learners". In: *arXiv preprint arXiv:2205.09113*.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He (2019a). "SlowFast Networks for Video Recognition". In: *Proc. ICCV*.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He (2019b). "SlowFast Networks for Video Recognition". In: *Proc. ICCV*.

Christoph Feichtenhofer, Haoqi Fan, Bo Xiong, Ross Girshick, and Kaiming He (2021). "A Large-Scale Study on Unsupervised Spatiotemporal Representation Learning". In: *Proc. CVPR*.

Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman (2016). "Convolutional Two-Stream Network Fusion for Video Action Recognition". In: *Proc. CVPR*.

Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould (2017). "Self-Supervised Video Representation Learning With Odd-One-Out Networks". In: *Proc. ICCV*.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov (2013). "DeViSE: A Deep Visual-Semantic Embedding Model". In: *NeurIPS*.

Valentin Gabeur, Chen Sun, Karteek Alahari, and Cordelia Schmid (2020). "Multi-modal Transformer for Video Retrieval". In: *Proc. ECCV*.

Chuang Gan, Boqing Gong, Kun Liu, Hao Su, and Leonidas J Guibas (2018). "Geometry Guided Convolutional Neural Networks for Self-Supervised Video Representation Learning". In: *Proc. CVPR*.

Chuang Gan, Tianbao Yang, and Boqing Gongi (2016a). "Learning Attributes Equals Multi-Source Domain Generalization". In: *Proc. CVPR*.

Chuang Gan, Yi Yang, Linchao Zhu, Deli Zhao, and Yueting Zhuang (2016b). "Recognizing an Action Using Its Name: A Knowledge-Based Approach". In: *IJCV*.

Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao (2021). "CLIP-Adapter: Better Vision-Language Models with Feature Adapters". In: *arXiv preprint arXiv:2110.04544*.

Tianyu Gao, Adam Fisch, and Danqi Chen (2021). "Making Pre-trained Language Models Better Few-shot Learners". In: *Proc. ACL*.

Zhanning Gao, Le Wang, Qilin Zhang, Zhenxing Niu, Nanning Zheng, and Gang Hua (2019). "Video imprint segmentation for temporal action detection in untrimmed videos". In: *Proc. AAAI*.

Yuying Ge, Yixiao Ge, Xihui Liu, Dian Li, Ying Shan, Xiaohu Qie, and Ping Luo (2022). "Bridging Video-Text Retrieval With Multiple Choice Questions". In: *Proc. CVPR*.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis (2018). "Unsupervised Representation Learning by Predicting Image Rotations". In: *Proc. ICLR*.

Yunchao Gong, Qifa Ke, and Michael Isardand Svetlana Lazebnik (2014a). "A multi-view embedding space for modeling internet images, tags, and their semantics". In: *IJCV*.

Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik (2014b). "Improving image-sentence embeddings using large weakly annotated photo collections". In: *Proc. ECCV*.

Melvyn A Goodale and A David Milner (1992). "Separate visual pathways for perception and action". In: *Trends in Neurosciences*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative adversarial nets". In: *NeurIPS*.

Alex Graves, Greg Wayne, and Ivo Danihelka (2014). "Neural turing machines". In: *arXiv preprint arXiv:1410.5401*.

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi (2022). "Kubric: a scalable dataset generator". In: *Proc. CVPR*.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko (2020). "Bootstrap your own latent: A new approach to self-supervised Learning". In: *NeurIPS*.

Agrim Gupta, Stephen Tian, Yunzhi Zhang, Jiajun Wu, Roberto Martín-Martín, and Li Fei-Fei (2022). "MaskViT: Masked Visual Pre-Training for Video Prediction". In: *arXiv preprint arXiv:2206.11894*.

Sumanth Gurram, David Chan, Andy Fang, and John Canny (n.d.). "LAVA: Language Audio Vision Alignment for Data-Efficient Video Pre-Training". In: *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith (2020). "Don't stop pretraining: adapt language models to domains and tasks". In: *arXiv preprint arXiv:2004.10964*.

Michael U. Gutmann and Aapo Hyvärinen (2010). “Noise-contrastive Estimation: A New Estimation Principle for Unnormalized Statistical Models”. In: *AISTATS*.

David Ha, Andrew Dai, and Quoc Le (2016). “HyperNetworks”. In: *Proc. ICLR*.

Tengda Han, Weidi Xie, and Andrew Zisserman (2019). “Video Representation Learning by Dense Predictive Coding”. In: *Workshop on Large Scale Holistic Video Understanding, ICCV*.

Tengda Han, Weidi Xie, and Andrew Zisserman (2020a). “Memory-augmented Dense Predictive Coding for Video Representation Learning”. In: *Proc. ECCV*.

Tengda Han, Weidi Xie, and Andrew Zisserman (2020b). “Self-supervised Co-training for Video Representation Learning”. In: *NeurIPS*.

Tengda Han, Weidi Xie, and Andrew Zisserman (2022). “Temporal Alignment Networks for Long-term Video”. In: *Proc. CVPR*.

Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh (2018). “Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?” In: *Proc. CVPR*.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick (2022). “Masked Autoencoders Are Scalable Vision Learners”. In: *Proc. CVPR*.

Kaiming He, Haoqi Fan, Auxin Wu, Saining Xie, and Ross Girshick (2020). “Momentum Contrast for Unsupervised Visual Representation Learning”. In: *Proc. CVPR*.

Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles (2015). “ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding”. In: *Proc. CVPR*.

Olivier J. Hénaff, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aäron van den Oord (2019). “Data-Efficient Image Recognition with Contrastive Predictive Coding”. In: *arXiv preprint arXiv:1905.09272*.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio (2019). “Learning deep representations by mutual information estimation and maximization”. In: *Proc. ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation*.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam (2017). "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *arXiv preprint arXiv:1704.04861*.

De-An Huang, Fei-Fei Li, and Juan Carlos Niebles (2016). "Connectionist Temporal Modeling for Weakly Supervised Action Labeling". In: *Proc. ECCV*.

Gabriel Huang, Bo Pang, Zhenhai Zhu, Clara Rivera, and Radu Soricut (2020). "Multimodal pretraining for dense video captioning". In: *arXiv preprint arXiv:2011.11760*.

Qingqiu Huang, Wentao Liu, and Dahua Lin (2018). "Person Search in Videos with One Portrait Through Visual and Temporal Links". In: *Proc. ECCV*. Springer-Verlag.

Ting-Hao Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. (2016). "Visual storytelling". In: *Proc. ACL*.

Noureldien Hussein, Efstratios Gavves, and Arnold W.M. Smeulders (2019a). "Timeception for Complex Action Recognition". In: *Proc. CVPR*.

Noureldien Hussein, Efstratios Gavves, and Arnold W.M. Smeulders (2019b). "VideoGraph: Recognizing Minutes-Long Human Activities in Videos". In: *ICCV-Workshop*.

Frank Hutter Ilya Loshchilov (2019). "Decoupled Weight Decay Regularization". In: *Proc. ICLR*.

Sergey Ioffe and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proc. ICML*.

Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson (2015). "Learning visual groups from co-occurrences in space and time". In: *Proc. ICLR*.

Allan Jabri, Andrew Owens, and Alexei Efros (2020). "Space-time correspondence as a contrastive random walk". In: *NeurIPS*.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira (2021). "Perceiver: General Perception with Iterative Attention". In: *Proc. ICML*.

Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi (2018). "Unsupervised Learning of Object Landmarks through Conditional Image Generation". In: *NeurIPS*.

Dinesh Jayaraman and Kristen Grauman (2015). "Learning image representations tied to ego-motion". In: *Proc. ICCV*.

Dinesh Jayaraman and Kristen Grauman (2016). "Slow and steady feature analysis: higher order temporal coherence in video". In: *Proc. CVPR*.

Xu Ji, João F. Henriques, and Andrea Vedaldi (2019). "Invariant Information Clustering for Unsupervised Image Classification and Segmentation". In: *Proc. ICCV*.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yunhsuan Sung, Zhen Li, and Tom Duerig (2021). "Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision". In: *Proc. ICML*.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim (2022). "Visual Prompt Tuning". In: *Proc. ECCV*.

Yu-Gang Jiang, Jingen Liu, A Roshan Zamir, George Toderici, Ivan Laptev, Mubarak Shah, and Rahul Sukthankar (2014). *THUMOS Challenge: Action Recognition with a Large Number of Classes*. http://crcv.ucf.edu/THUMOS14/.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig (2020). "How Can We Know What Language Models Know?" In: *Transactions of the Association for Computational Linguistics*.

Longlong Jing and Yingli Tian (2018). "Self-supervised Spatiotemporal Feature Learning by Video Geometric Transformations". In: *arXiv preprint arXiv:1811.11387*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu (2016). "Exploring the limits of language modeling". In: *arXiv preprint arXiv:1602.02410*.

Chen Ju, Tengda Han, Kunhao Zheng, Ya Zhang, and Weidi Xie (2022). "Prompting Visual-Language Models for Efficient Video Understanding". In: *Proc. ECCV*.

Chen Ju, Peisen Zhao, Siheng Chen, Ya Zhang, Yanfeng Wang, and Qi Tian (2021). "Divide and Conquer for Single-Frame Temporal Action Localization". In: *Proc. ICCV*.

Andrej Karpathy and Li Fei-Fei (2015). "Deep visual-semantic alignments for generating image descriptions". In: *Proc. CVPR*.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret (2020). "Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention". In: *Proc. ICML*.

Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman (2017). "The Kinetics Human Action Video Dataset." In: *arXiv preprint arXiv:1705.06950*.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan (2020). "Supervised Contrastive Learning". In: *arXiv preprint arXiv:2004.11362*.

Dahun Kim, Donghyeon Cho, and In So Kweon (2019). "Self-Supervised Video Representation Learning with Space-Time Cubic Puzzles". In: *Proc. AAAI*.

Diederik P. Kingma and Jimmy Ba (2014a). "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980*.

Diederik P. Kingma and Jimmy Ba (2014b). "Adam: A Method for Stochastic Optimization". In: *CoRR*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel (2014). "Unifying visual-semantic embeddings with multimodal neural language models". In: *arXiv preprint arXiv:1411.2539*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya (2020). "Reformer: The Efficient Transformer". In: *Proc. ICLR*.

Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf (2015). "Associating neural word embeddings with deep image represen- tations using fisher vectors". In: *Proc. CVPR*.

Bruno Korbar, Du Tran, and Lorenzo Torresani (2018). "Cooperative Learning of Audio and Video Models from Self-Supervised Synchronization". In: *NeurIPS*.

Bruno Korbar, Du Tran, and Lorenzo Torresani (2019). “SCSampler: Sampling Salient Clips from Video for Efficient Action Recognition”. In: *Proc. ICCV*.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles (2017a). “Dense-captioning events in videos”. In: *Proc. ICCV*.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles (2017b). “Dense-Captioning Events in Videos”. In: *Proc. ICCV*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks”. In: *NeurIPS*.

Hilde Kuehne, Ali Arslan, and Thomas Serre (2014). “The Language of Actions: Recovering the Syntax and Semantics of Goal-Directed Human Activities”. In: *Proc. CVPR*.

Hilde Kuehne, A. Richard, and Juergen Gall (2017). “Weakly supervised learning of actions from transcripts”. In: *CVIU*.

Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre (2011). “HMDB: A large video database for human motion recognition”. In: *Proc. ICCV*.

Anna Kukleva, Makarand Tapaswi, and Ivan Laptev (June 2020). “Learning Interactions and Relationships Between Movie Characters”. In: *Proc. CVPR*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher (2016). “Ask me anything: Dynamic memory networks for natural language processing”. In: *Proc. ICML*.

Zihang Lai, Erika Lu, and Weidi Xie (2020). “MAST: A Memory-augmented Self-supervised Tracker”. In: *Proc. CVPR*.

Zihang Lai and Weidi Xie (2019). “Self-supervised Learning for Video Correspondence Flow”. In: *Proc. BMVC*.

Guillaume Lample, Alexandre Sablayrolles, Marc’Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou (2019). “Large Memory Layers with Product Keys”. In: *NeurIPS*.

I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld (2008). “Learning realistic human actions from movies”. In: *Proc. CVPR*.

Victor Lavrenko, Raghavan Manmatha, and Jiwoon Jeon (2003). “A model for learning the semantics of pictures”. In: *NeurIPS*.

Hei Law and Jia Deng (2018). “CornerNet: Detecting objects as paired keypoints”. In: *Proc. ECCV*.

Colin Lea, Michael D. Flynn, René Vidal, Austin Reiter, and Gregory D. Hager (2017). “Temporal Convolutional Networks for Action Segmentation and Detection”. In: *Proc. CVPR*.

Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang (2017). “Unsupervised Representation Learning by Sorting Sequence”. In: *Proc. ICCV*.

Jie Lei, Linjie Li, Luowei Zhou, Zhe Gan, Tamara L. Berg, Mohit Bansal, and Jingjing Liu (2021). “Less is More: ClipBERT for Video-and-Language Learningvia Sparse Sampling”. In: *Proc. CVPR*.

Peng Lei and Sinisa Todorovic (2018). “Temporal deformable residual networks for action segmentation in videos”. In: *Proc. CVPR*.

Brian Lester, Rami Al-Rfou, and Noah Constant (2021). “The Power of Scale for Parameter-Efficient Prompt Tuning”. In: *Proc. EMNLP*.

Gen Li, Nan Duan, Yuejian Fang, Ming Gong, Daxin Jiang, and Ming Zhou (2020). “Unicoder-VL: A Universal Encoder for Vi- sion and Language by Cross-modal Pre-training”. In: *Proc. AAAI*.

Jun Li, Peng Lei, and Sinisa Todorovic (2019). “Weakly Supervised Energy-Based Learning for Action Segmentation”. In: *Proc. ICCV*.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi (2022). “BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation”. In: *Proc. ICML*.

Junnan Li, Ramprasaath R. Selvaraju, Akhilesh Deepak Gotmare, Shafiq Joty, Caiming Xiong, and Steven Hoi (2021). “Align before Fuse: Vision and Language Representation Learning with Momentum Distillation”. In: *NeurIPS*.

Junnan Li, Yongkang Wong, Qi Zhao, and M. Kankanhalli (2020). “Video Storytelling: Textual Summaries for Events”. In: *IEEE Transactions on Multimedia*.

Xiang Lisa Li and Percy Liang (2021). “Prefix-Tuning: Optimizing Continuous Prompts for Generation”. In: *Proc. ACL*.

Yehao Li, Ting Yao, Yingwei Pan, Hongyang Chao, and Tao Mei (2018). "Jointly localizing and describing events for dense video captioning". In: *Proc. CVPR*.

Yikang Li, Sheng-hung Hu, and Baoxin Li (2016). "Recognizing unseen actions in a domain-adapted embedding space". In: *Intl. Conf. Image Proc.*

Feng Liang, Yangguang Li, and Diana Marculescu (2022). "SupMAE: Supervised Masked Autoencoders Are Efficient Vision Learners". In: *arXiv preprint arXiv:2205.14540*.

Chin-Yew Lin (2004). "Rouge: A package for automatic evaluation of summaries". In: *Text summarization branches out*.

Chuming Lin, Chengming Xu, Donghao Luo, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yanwei Fu (2021). "Learning Salient Boundary Feature for Anchor-free Temporal Action Localization". In: *Proc. CVPR*.

Ji Lin, Chuang Gan, and Song Han (2019). "TSM: Temporal shift module for efficient video understanding". In: *Proc. ICCV*.

Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen (2019). "BMN: Boundary-matching network for temporal action proposal generation". In: *Proc. ICCV*.

Tianwei Lin, Xu Zhao, and Zheng Shou (2017). "Single shot temporal action detection". In: *Proc. ACMMM*.

Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang (2018). "BSN: Boundary Sensitive Network for Temporal Action Proposal Generation". In: *Proc. ECCV*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár (2017). "Focal loss for dense object detection". In: *Proc. ICCV*.

Xudong Lin, Fabio Petroni, Gedas Bertasius, Marcus Rohrbach, Shih-Fu Chang, and Lorenzo Torresani (2022). "Learning To Recognize Procedural Activities with Distant Supervision". In: *Proc. CVPR*.

Jingen Liu, Benjamin Kuipers, and Silvio Savarese (2011). "Recognizing Human Actions by Attributes". In: *Proc. CVPR*.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg (2016). “SSD: Single shot multibox detector”. In: *Proc. ECCV*. Springer.

Yang Liu, Samuel Albanie, Arsha Nagrani, and Andrew Zisserman (2019). “Use what you have: Video retrieval using representations from collaborative experts”. In: *Proc. BMVC*.

Yue Liu, Christos Matsoukas, Fredrik Strand, Hossein Azizpour, and Kevin Smith (2022). “PatchDropout: Economizing Vision Transformers Using Patch Dropout”. In: *arXiv preprint arXiv:2208.07220*.

William Lotter, Gabriel Kreiman, and David Cox (2017). “Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning”. In: *Proc. ICLR*.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee (2019). “Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks”. In: *NeurIPS*.

Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee (2020). “12-in-1: Multi-task vision and language representation learning”. In: *Proc. CVPR*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh (2018). “Neural Baby Talk”. In: *Proc. CVPR*.

Dezhao Luo, Chang Liu, Yu Zhou, Dongbao Yang, Can Ma, Qixiang Ye, and Weiping Wang (2020). “Video Cloze Procedure for Self-Supervised Spatio-Temporal Learning”. In: *Proc. AAAI*.

Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Jason Li, Taroon Bharti, and Ming Zhou (2020). “Univl: A unified video and language pre-training model for multimodal understanding and generation”. In: *arXiv preprint arXiv:2002.06353*.

Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li (2021). “CLIP4Clip: An Empirical Study of CLIP for End to End Video Clip Retrieval”. In: *arXiv preprint arXiv:2104.08860*.

Richard Marquand (1983). *Star Wars: Episode VI – Return of the Jedi*. 20th Century Fox.

Marcin Marszałek, Ivan Laptev, and Cordelia Schmid (2009). “Actions in Context”. In: *Proc. CVPR*.

Michael Mathieu, Camille Couprie, and Yann LeCun (2016). “Deep Multi-scale Video Prediction Beyond Mean Square Error.” In: *ICLR*.

Pascal Mettes, William Thong, and Cees G. M. Snoek (2021). “Object Priors for Classifying and Localizing Unseen Actions”. In: *IJCV*.

Antoine Miech, Jean-Baptiste Alayrac, Ivan Laptev, Josef Sivic, and Andrew Zisserman (2021). “Thinking Fast and Slow: Efficient Text-to-Visual Retrieval with Transformers”. In: *Proc. CVPR*.

Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman (2020). “End-to-End Learning of Visual Representations from Uncurated Instructional Videos”. In: *Proc. CVPR*.

Antoine Miech, Ivan Laptev, and Josef Sivic (2018). “Learning a Text-Video Embedding from Imcomplete and Heterogeneous Data”. In: *arXiv preprint arXiv:1804.02516*.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic (2019). “HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips”. In: *Proc. ICCV*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean (2013a). “Efficient Estimation of Word Representations in Vector Space”. In: *NeurIPS*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean (2013b). “Distributed Representations of Words and Phrases and their Compositionality”. In: *NeurIPS*.

Ashish Mishra, Anubha Pandey, and Hema A. Murthy (2020). “Zero-shot learning for action recognition using synthesized features”. In: *Neurocomputing*.

Ishan Misra and Laurens van der Maaten (2020). “Self-Supervised Learning of Pretext-Invariant Representations”. In: *Proc. CVPR*.

Ishan Misra, C. Lawrence Zitnick, and Martial Hebert (2016). “Shuffle and Learn: Unsupervised Learning using Temporal Order Verification”. In: *Proc. ECCV*.

Andriy Mnih and Koray Kavukcuoglu (2013). “Learning word embeddings efficiently with noise-contrastive estimation”. In: *NeurIPS*.

Ron Mokady, Amir Hertz, and Amit H Bermano (2021). “ClipCap: CLIP prefix for image captioning”. In: *arXiv preprint arXiv:2111.09734*.

Mathew Monfort, SouYoung Jin, Alexander Liu, David Harwath, Rogerio Feris, James Glass, and Aude Oliva (2021a). "Spoken moments: Learning joint audio-visual representations from video descriptions". In: *Proc. CVPR*.

Mathew Monfort, Bowen Pan, Kandan Ramakrishnan, Alex Andonian, Barry A McNamara, Alex Lascelles, Quanfu Fan, Dan Gutfreund, Rogerio Feris, and Aude Oliva (2021b). "Multi-moments in time: Learning and interpreting models for multi-action video understanding". In: *IEEE PAMI*.

Pedro Morgado, Nuno Vasconcelos, and Ishan Misra (2021). "Audio-visual instance discrimination with cross-modal agreement". In: *Proc. CVPR*.

Yasuhide Mori, Hironobu Takahashi, and Ryuichi Oka (1999). "Image-to-word transformation based on dividing and vector quantizing images with words". In: *MISRM*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen (2016). "A corpus and cloze evaluation for deeper understanding of commonsense stories". In: *Proc. ACL*.

Jonghwan Mun, Linjie Yang, Zhou Ren, Ning Xu, and Bohyung Han (2019). "Streamlined dense video captioning". In: *Proc. CVPR*.

Arsha Nagrani, Paul Hongsuck Seo, Bryan Seybold, Anja Hauth, Santiago Manen, Chen Sun, and Cordelia Schmid (2022). "Learning Audio-Video Modalities from Image Captions". In: *Proc. ECCV*.

Megha Nawhal and Greg Mori (2021). "Activity Graph Transformer for Temporal Action Localization". In: *arXiv preprint arXiv:2101.08540*.

John Pringle Nichol (1857). *A Cyclopedia of the Physical Sciences*. Richard Griffin and Company.

Mehdi Noroozi and Paolo Favaro (2016). "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles". In: *Proc. ECCV*.

Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro (2017). "Representation learning by learning to count". In: *Proc. ICCV*.

David Nukrai, Ron Mokady, and Amir Globerson (2022). "Text-Only Training for Image Captioning using Noise-Injected CLIP". In: *arXiv preprint arXiv:2211.00575*.

Andreea-Maria Oncescu, João F. Henriques, Yang Liu, Andrew Zisserman, and Samuel Albanie (2021). "QuerYD: A video dataset with high-quality textual and audio narrations". In: *Proc. ICASSP*.

Yingwei Pan, Tao Mei, Ting Yao, Houqiang Li, and Yong Rui (2016). "Jointly modeling embedding and translation to bridge video and language". In: *Proc. CVPR*.

Pinelopi Papalampidi, Frank Keller, and Mirella Lapata (2019). "Movie plot analysis via turning point identification". In: *arXiv preprint arXiv:1908.10328*.

Jae Sung Park, Marcus Rohrbach, Trevor Darrell, and Anna Rohrbach (2019). "Adversarial Inference for Multi-Sentence Video Description". In: *Proc. CVPR*.

Deepak Pathak, Ross B. Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan (2017). "Learning features by watching objects move." In: *Proc. CVPR*.

Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros (2016). "Context encoders: Feature learning by inpainting." In: *Proc. CVPR*.

Mandela Patrick, Yuki M. Asano, Ruth Fong, João F. Henriques, Geoffrey Zweig, and Andrea Vedaldi (2020). "Multi-modal Self-Supervision from Generalized Data Transformations". In: *arXiv preprint arXiv:2003.04298*.

Mandela Patrick, Yuki M. Asano, Polina Kuznetsova, Ruth Fong, João F. Henriques, Geoffrey Zweig, and Andrea Vedaldi (2021a). "On Composition of Transformations in Contrastive Self-Supervised Learning". In: *Proc. ICCV*.

Mandela Patrick, Dylan Campbell, Yuki M. Asano, Ishan Misra, Florian Metze, Christoph Feichtenhofer, Andrea Vedaldi, and João F. Henriques (2021b). "Keeping Your Eye on the Ball: Trajectory Attention in Video Transformers". In: *NeurIPS*.

Alonso Patron-Perez, M. Marszałek, Andrew Zisserman, and Ian D. Reid (2010). "High Five: Recognising Human Interactions in TV Shows". In: *Proc. BMVC*.

Georgios Pavlakos, Ethan Weber, Matthew Tancik, and Angjoo Kanazawa (2022). "The One Where They Reconstructed 3D Humans and Environments in TV Shows". In: *Proc. ECCV*. Ed. by Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner.

Toby Perrett, Alessandro Masullo, Tilo Burghardt, Majid Mirmehdi, and Dima Damen (2021). "Temporal Relational CrossTransformers for Few-Shot Action Recognition". In: *Proc. CVPR*.

Jean Piaget (1977). *The essential Piaget*. Ed. by Howard E Gruber and J Jacques Vonèche. Routledge & Kegan Paul London.

AJ Piergiovanni, Anelia Angelova, and Michael S. Ryoo (2020). "Evolving Losses for Unsupervised Video Representation Learning". In: *Proc. CVPR*.

Bryan A Plummer, Matthew Brown, and Svetlana Lazebnik (2017). "Enhancing video summarization via vision-language embedding". In: *Proc. CVPR*.

Jean Baptiste Polle (n.d.). *Camembert-NER: model fine-tuned from camemBERT for NER task*. https://huggingface.co/Jean-Baptiste/camembert-ner. Accessed: 2022-11-01.

Rui Qian, Shuangrui Ding, Xian Liu, and Dahua Lin (2022a). "Static and Dynamic Concepts for Self-supervised Video Representation Learning". In: *arXiv preprint arXiv:2207.12795*.

Rui Qian, Yeqing Li, Zheng Xu, Ming-Hsuan Yang, Serge Belongie, and Yin Cui (2022b). "Multimodal Open-Vocabulary Video Classification via Pre-Trained Vision and Language Models". In: *arXiv preprint arXiv:2207.07646*.

Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui (2020). "Spatiotemporal Contrastive Video Representation Learning". In: *arXiv preprint arXiv:2008.03800*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever (2021). "Learning Transferable Visual Models From Natural Language Supervision". In: *Proc. ICML*.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever (2022). "Robust Speech Recognition via Large-Scale Weak Supervision". In: *OpenAI blog*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). "Language Models are Unsupervised Multitask Learners". In:

Tanzila Rahman, Bicheng Xu, and Leonid Sigal (2019). "Watch, listen and tell: Multi-modal weakly supervised dense event captioning". In: *Proc. ICCV*.

Rajesh P. N. Rao and Dana H. Ballard (1999). "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects". In: *Nature Neuroscience*.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi (2016). "XNOR-Net: Imagenet classification using binary convolutional neural networks". In: *Proc. ECCV*.

Hareesh Ravi, Kushal Kafle, Scott Cohen, Jonathan Brandt, and Mubbasir Kapadia (2021). "AESOP: Abstract Encoding of Stories, Objects, and Pictures". In: *Proc. ICCV*.

Adria Recasens, Pauline Luc, Jean-Baptiste Alayrac, Luyu Wang, Florian Strub, Corentin Tallec, Mateusz Malinowski, Viorica Pătrăucean, Florent Altché, Michal Valko, et al. (2021). "Broaden your views for self-supervised video learning". In: *Proc. ICCV*.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi (2016). "You Only Look Once: Unified, Real-Time Object Detection". In: *Proc. CVPR*.

Video Description Research and Development Center (2013). *YouDescribe*. URL: https://youdescribe.org/.

Alexander Richard, Hilde Kuehne, Ahsan Iqbal, and Juergen Gall (2018). "NeuralNetwork-Viterbi: A Framework for Weakly Supervised Video Learning". In: *Proc. CVPR*.

Anna Rohrbach, Marcus Rohrbach, Niket Tandon, and Bernt Schiele (2015). "A dataset for movie description". In: *Proc. CVPR*.

Anna Rohrbach, Atousa Torabi, Marcus Rohrbach, Niket Tandon, Christopher Pal, Hugo Larochelle, Aaron Courville, and Bernt Schiele (2017). "Movie description". In: *IJCV*.

Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele (2012). "A Database for Fine Grained Activity Detection of Cooking Activities". In: *Proc. CVPR*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier (2020). "Efficient Content-Based Sparse Attention with Routing Transformers". In: *TACL*.

K. Pramod Sankar, Saurabh Pandey, and C. V. Jawahar (2006). "Text Driven Temporal Segmentation of Cricket Videos". In: *Indian Conference on Computer Vision, Graphics and Image Processing*.

Pramod Sankar, C. V. Jawahar, and Andrew Zisserman (2009). "Subtitle-Free Movie to Script Alignment". In: *Proc. BMVC*.

Timo Schick and Hinrich Schütze (2021). “Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference”. In:

Paul Hongsuck Seo, Arsha Nagrani, Anurag Arnab, and Cordelia Schmid (2022). “End-to-end generative pretraining for multimodal video captioning”. In: *Proc. CVPR*.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut (2018). “Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning”. In: *Proc. ACL*.

Zhiqiang Shen, Jianguo Li, Zhou Su, Minjun Li, Yurong Chen, Yu-Gang Jiang, and Xiangyang Xue (2017). “Weakly supervised dense video captioning”. In: *Proc. CVPR*.

Botian Shi, Lei Ji, Yaobo Liang, Nan Duan, Peng Chen, Zhendong Niu, and Ming Zhou (2019). “Dense procedure captioning in narrated instructional videos”. In: *Proc. ACL*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh (2020). “AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts”. In: *Proc. EMNLP*.

Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang (2017). “CDC: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos”. In: *Proc. CVPR*.

Zheng Shou, Dongang Wang, and Shih-Fu Chang (2016). “Temporal Action Localization in Untrimmed Videos via Multi-stage CNNs”. In: *Proc. CVPR*.

Gunnar A. Sigurdsson, Gül Varol, X. Wang, Ali Farhadi, I. Laptev, and A. Gupta (2016). “Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding”. In: *Proc. ECCV*.

Karen Simonyan and Andrew Zisserman (2014). “Two-Stream Convolutional Networks for Action Recognition in Videos”. In: *NeurIPS*.

Bharat Singh, Tim K Marks, Michael Jones, Oncel Tuzel, and Ming Shao. (2016). “A multi-stream bi-directional recurrent neural network for fine-grained action detection”. In: *Proc. CVPR*.

Linda Smith, Alexandra Quittner, Mary Osberger, and Richard Miyamoto (1998). "Audition and visual attention: The developmental trajectory in deaf and hearing populations". In: *Developmental psychology*.

Mattia Soldan, Alejandro Pardo, Juan León Alcázar, Fabian Caba, Chen Zhao, Silvio Giancola, and Bernard Ghanem (2022). "MAD: A scalable dataset for language grounding in videos from movie audio descriptions". In: *Proc. CVPR*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu (2020). "MPNet: Masked and Permuted Pre-training for Language Understanding". In: *arXiv preprint arXiv:2004.09297*.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah (2012). "UCF101: A Dataset Of 101 Human Actions Classes From Videos In The Wild". In: *arXiv preprint arXiv:1212.0402*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *JMLR*.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov (2015). "Unsupervised Learning of Video Representations using LSTMs". In: *Proc. ICML*.

Jonathan C. Stroud, David A. Ross, Chen Sun, Jia Deng, and Rahul Sukthankar (2018). "D3D: Distilled 3D Networks for Video Action Recognition". In: *arXiv preprint arXiv:1812.08249*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum (2019). "Energy and policy considerations for deep learning in NLP". In: *Proc. ACL*.

Haisheng Su, Weihao Gan, Wei Wu, Yu Qiao, and Junjie Yan (2021). "Bsn++: Complementary boundary regressor with scale-balanced relation modeling for temporal action proposal generation". In: *Proc. AAAI*.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai (2019). "Vl-bert: Pre-training of generic visual- linguistic representations". In: *Proc. ICLR*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus (2015). "End-to-end memory networks". In: *NeurIPS*.

Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid (2019). "Contrastive Bidirectional Transformer for Temporal Representation Learning". In: *arXiv preprint arXiv:1906.05743*.

Yidan Sun, Qin Chao, and Boyang Li (2022). “Synopses of Movie Narratives: a Video-Language Dataset for Story Understanding”. In: *arXiv preprint arXiv:2203.05711*.

Dídac Surís, Dave Epstein, and Carl Vondrick (2022). “Globetrotter: Connecting Languages by Connecting Images”. In: *Proc. CVPR*.

Dídac Surís, Ruoshi Liu, and Carl Vondrick (2021). “Learning the Predictability of the Future”. In: *Proc. CVPR*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le (2014). “Sequence to sequence learning with neural networks”. In: *NeurIPS*.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour (2000). “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. In: *NeurIPS*.

Oren Tadmor, Yonatan Wexler, Tal Rosenwein, Shai Shalev-Shwartz, and Amnon Shashua (2016). “Learning a Metric Embedding for Face Recognition using the Multibatch Method”. In: *NeurIPS*.

Hao Tan and Mohit Bansal (2019). “Lxmert: Learning cross- modality encoder representations from transformers”. In: *Proc. EMNLP*.

Jing Tan, Jiaqi Tang, Limin Wang, and Gangshan Wu (2021). “Relaxed transformer decoders for direct action proposal generation”. In: *Proc. ICCV*.

Mingxing Tan and Quoc V. Le (2019). “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *Proc. ICML*.

Mingkang Tang, Zhanyu Wang, Zhenhua Liu, Fengyun Rao, Dian Li, and Xiu Li (2021). “Clip4caption: Clip for video caption”. In: *Proc. ACMMM*.

Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou (2019). “COIN: A Large-scale Dataset for Comprehensive Instructional Video Analysis”. In: *Proc. CVPR*.

M. Tapaswi, M. Baeuml, and R. Stiefelhagen (2012). ““Knock! Knock! Who is it?” Probabilistic Person Identification in TV Series”. In: *Proc. CVPR*.

Makarand Tapaswi, Martin Bauml, and Rainer Stiefelhagen (2015a). “Book2movie: Aligning video scenes with book chapters”. In: *Proc. CVPR*.

Makarand Tapaswi, Martin Bäuml, and Rainer Stiefelhagen (2015b). "Aligning Plot Synopses to Videos for Story-based Retrieval". In: *International Journal of Multimedia Information Retrieval*.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan (2020). "Sparse Sinkhorn Attention". In: *arXiv preprint arXiv:2002.11296*.

Yonglong Tian, Dilip Krishnan, and Phillip Isola (2020a). "Contrastive Multiview Coding". In: *Proc. ECCV*.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola (2020b). "What makes for good views for contrastive learning?" In: *NeurIPS*.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang (2022). "VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training". In: *arXiv preprint arXiv:2203.12602*.

Atousa Torabi, Christopher Pal, Hugo Larochelle, and Aaron Courville (2015). "Using descriptive video services to create a large data source for video annotation research". In: *arXiv preprint arXiv:1503.01070*.

Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri (2018). "A Closer Look at Spatiotemporal Convolutions for Action Recognition". In: *Proc. CVPR*.

Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill (2021). "Multimodal few-shot learning with frozen language models". In: *NeurIPS*.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky (2016). "Instance Normalization: The Missing Ingredient for Fast Stylization". In: *arXiv preprint arXiv:1607.08022*.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals (2018). "Representation Learning with Contrastive Predictive Coding". In: *arXiv preprint arXiv:1807.03748*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *NeurIPS*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh (2015). "Cider: Consensus-based image description evaluation". In: *Proc. CVPR*.

Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko (2015). "Sequence to Sequence – Video to Text". In: *Proc. ICCV*.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba (2016a). "Anticipating Visual Representations From Unlabelled Video." In: *Proc. CVPR*.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba (2016b). "Generating Videos with Scene Dynamics". In: *NeurIPS*.

Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy (2018). "Tracking Emerges by Colorizing Videos". In: *Proc. ECCV*.

Hermann von Helmholtz (1867). *Treatise on Physiological Optics*.

Jiangliu Wang, Jianbo Jiao, and Yun-Hui Liu (2020). "Self-supervised Video Representation Learning by Pace Prediction". In: *Proc. ECCV*.

Jingwen Wang, Wenhao Jiang, Lin Ma, Wei Liu, and Yong Xu (2018). "Bidirectional attentive fusion with context gating for dense video captioning". In: *Proc. CVPR*.

Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Val Gool (2016). "Temporal Segment Networks: Towards Good Practices for Deep Action Recognition". In: *Proc. ECCV*.

Mengmeng Wang, Jiazheng Xing, and Yong Liu (2021). "ActionCLIP: A New Paradigm for Video Action Recognition". In: *arXiv preprint arXiv:2109.08472*.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma (2020). "Linformer: Self-Attention with Linear Complexity". In: *arXiv preprint arXiv:2006.04768*.

Teng Wang, Ruimao Zhang, Zhichao Lu, Feng Zheng, Ran Cheng, and Ping Luo (2021). "End-to-end dense video captioning with parallel decoding". In: *Proc. ICCV*.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He (2018). "Non-local Neural Networks." In: *Proc. CVPR*.

Xiaolong Wang and Abhinav Gupta (2015). "Unsupervised Learning of Visual Representations using Videos". In: *Proc. ICCV*.

Xiaolong Wang, Allan Jabri, and Alexei A. Efros (2019). "Learning Correspondence from the Cycle-Consistency of Time". In: *Proc. CVPR*.

Yujia Wang, Wei Liang, Haikun Huang, Yongqi Zhang, Dingzeyu Li, and Lap-Fai Yu (2021). "Toward automatic audio description generation for accessible videos". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*.

Andrew B. Watson (1986). "Temporal Sensitivity". In: *Sensory Processes and Perception*.

Donglai Wei, Joseph Lim, Andrew Zisserman, and William T. Freeman (2018). "Learning and Using the Arrow of Time". In: *Proc. CVPR*.

Jason Weston, Samy Bengio, and Nicolas Usunier (2011). "WSABIE: Scaling Up to Large Vocabulary Image Annotation". In: *Proc. IJCAI*.

Olivia Wiles, A. Sophia Koepke, and Andrew Zisserman (2018). "Self-supervised learning of a facial attribute embedding from video". In: *Proc. BMVC*.

Laurenz Wiskott and Terrence Sejnowski (2002). "Slow feature analysis: Unsupervised learning of invariances". In: *Neural Computation*.

Chao-Yuan Wu, Ross Girshick, Kaiming He, Christoph Feichtenhofer, and Philipp Krähenbühl (2020). "A Multigrid Method for Efficiently Training Video Models". In: *Proc. CVPR*.

Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin (2018). "Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination". In: *Proc. CVPR*.

Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy (2018). "Rethinking Spatiotemporal Feature Learning For Video Understanding". In: *Proc. ECCV*.

Yu Xiong, Qingqiu Huang, Lingfeng Guo, Hang Zhou, Bolei Zhou, and Dahua Lin (2019). "A graph-based framework to bridge movies and synopses". In: *Proc. ICCV*.

Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang (2019). "Self-Supervised Spatiotemporal Learning via Video Clip Order Prediction". In: *Proc. CVPR*.

Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer (2021). "VideoCLIP: Contrastive Pre-training for Zero-shot Video-Text Understanding". In: *arXiv preprint arXiv:2109.14084*.

Huijuan Xu, Abir Das, and Kate Saenko (2017). "R-C3D: Region Convolutional 3D Network for Temporal Activity Detection". In: *Proc. ICCV*.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui (2016). "MSR-VTT: A large video description dataset for bridging video and language". In: *Proc. CVPR*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio (2015). "Show, attend and tell: Neural image caption generation with visual attention". In: *Proc. ICML*.

Mengmeng Xu, Chen Zhao, David S Rojas, Ali Thabet, and Bernard Ghanem (2020). "G-tad: Sub-graph localization for temporal action detection". In: *Proc. CVPR*.

Antoine Yang, Antoine Miech, Josef Sivic, Ivan Laptev, and Cordelia Schmid (2022). "Zero-Shot Video Question Answering via Frozen Bidirectional Language Models". In: *arXiv preprint arXiv:2206.08155*.

Jianwei Yang, Yonatan Bisk, and Jianfeng Gao (2021). "TACo: Token-aware Cascade Contrastive Learning for Video-Text Alignment". In: *Proc. ICCV*.

Le Yang, Houwen Peng, Dingwen Zhang, Jianlong Fu, and Junwei Han (2020). "Revisiting anchor mechanisms for temporal action localization". In: *IEEE Transactions on Image Processing*.

Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei (2016). "End-to-end learning of action detection from frame glimpses in videos". In: *Proc. CVPR*.

Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu (2022). "CoCa: Contrastive Captioners are Image-Text Foundation Models". In: *Transactions on Machine Learning Research*.

Youngjae Yu, Jiwan Chung, Heeseung Yun, Jongseok Kim, and Gunhee Kim (2021). "Transitional adaptation of pretrained models for visual storytelling". In: *Proc. CVPR*.

Youngjae Yu, Jongseok Kim, and Gunhee Kim (2018). "A joint sequence fusion model for video question answering and retrieval". In: *Proc. ECCV*.

Christopher Zach, Thomas Pock, and Horst Bischof (2007). "A Duality Based Approach for Realtime TV-L1 Optical Flow". In: *Pattern Recognition*.

Chuhan Zhang, Ankush Gupta, and Andrew Zisserman (2021). "Temporal Query Networks for Fine-grained Video Understanding". In: *Proc. CVPR*.

Hongguang Zhang, Li Zhang, Xiaojuan Qi, Hongdong Li, Philip H S Torr, and Piotr Koniusz (2020). “Few-shot Action Recognition with Permutation-invariant Attention”. In: *Proc. ECCV*.

Renrui Zhang, Rongyao Fang, Peng Gao, Wei Zhang, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li (2021). “Tip-Adapter: Training-free CLIP-Adapter for Better Vision-Language Modeling”. In: *arXiv preprint arXiv:2111.03930*.

Richard Zhang, Phillip Isola, and Alexei A. Efros (2016). “Colorful Image Colorization”. In: *Proc. ECCV*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi (2020). “Bertscore: Evaluating text generation with bert”. In: *Proc. ICLR*.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun (2018). “Shufflenet: An extremely efficient convolutional neural network for mobile devices”. In: *Proc. CVPR*.

Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma (2016). “Single-Image Crowd Counting via Multi-Column Convolutional Neural Network”. In: *Proc. CVPR*.

Hang Zhao, Antonio Torralba, Lorenzo Torresani, and Zhicheng Yan (2019). “Hacs: Human action clips and segments dataset for recognition and temporal localization”. In: *Proc. ICCV*.

Peisen Zhao, Lingxi Xie, Chen Ju, Ya Zhang, Yanfeng Wang, and Qi Tian (2020). “Bottom-up temporal action localization with mutual regularization”. In: *Proc. ECCV*.

Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin (2017). “Temporal Action Detection with Structured Segment Networks”. In: *Proc. ICCV*.

Zhenda, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu (2022). “SimMIM: A Simple Framework for Masked Image Modeling”. In: *Proc. CVPR*.

Jiaming Zhou, Kun-Yu Lin, Haoxin Li, and Wei-Shi Zheng (2021). “Graph-Based High-Order Relation Modeling for Long-Term Action Recognition”. In: *Proc. CVPR*.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu (2021). "Learning to prompt for vision-language models". In: *arXiv preprint arXiv:2109.01134*.

Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, and Jason J. Corso (2020). "Unified vision-language pre-training for image captioning and vqa". In: *Proc. AAAI*.

Luowei Zhou, Chenliang Xu, and Jason J Corso (2018a). "Towards Automatic Learning of Procedures From Web Instructional Videos". In: *Proc. AAAI*.

Luowei Zhou, Chenliang Xu, and Jason J Corso (2021). *YouCook2 Leaderboard*. https://github.com/LuoweiZhou/YouCook2-Leaderboard.

Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong (2018b). "End-to-end dense video captioning with masked transformer". In: *Proc. CVPR*.

Linchao Zhu and Yi Yang (2018). "Compound Memory Networks for Few-Shot Video Classification". In: *Proc. ECCV*.

Linchao Zhu and Yi Yang (2020a). "ActBERT: Learning Global-Local Video-Text Representations". In: *Proc. CVPR*.

Linchao Zhu and Yi Yang (2020b). "Label independent memory for semi-supervised few-shot video classification". In: *IEEE PAMI*.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler (2015). "Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books". In: *Proc. CVPR*.

Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins (2019). "Local Aggregation for Unsupervised Learning of Visual Embeddings". In: *Proc. ICCV*.

Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic (2019). "Cross-task Weakly Supervised Learning from Instructional Videos". In: *Proc. CVPR*.

Appendix A

Statement of Authorship

A statement of authorship is provided for each multi-authored paper included in this thesis. The statements describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication, there exists a complete statement that is filled out and signed by the candidate and supervisor.

Statement of Authorship for the paper "Video Representation Learning by Dense Predictive Coding" in Chapter 2.

Paper title	Video Representation Learning by Dense Predictive Coding
Authors	**Tengda Han**, Weidi Xie, Andrew Zisserman
Publication status	Published
Publication details	[Oral] Workshop on Large Scale Holistic Video Understanding, International Conference on Computer Vision (ICCV), 2019.

Student Confirmation

Student name	Tengda Han	
Contribution to the paper	First-author contribution: • conception of research ideas • design and implementation of models • running of large-scale experiments • writing and presentation of the paper	
Signature and Date		Sept. 29th 2022

Supervisor Confirmation

By signing the Statement of Authorship, the supervisor is certifying that the candidate made a substantial contribution to the publication, and that the description above is accurate.

Supervisor name	Prof. Andrew Zisserman	
Supervisor comments		
Signature and Date		Sept. 29th 2022

Statement of Authorship for the paper "Memory-augmented Dense Predictive Coding for Video Representation Learning" in Chapter 3.

Paper title	Memory-augmented Dense Predictive Coding for Video Representation Learning
Authors	**Tengda Han**, Weidi Xie, Andrew Zisserman
Publication status	Published
Publication details	[Spotlight] European Conference on Computer Vision (ECCV), 2020.

Student Confirmation

Student name	Tengda Han	
Contribution to the paper	First-author contribution: • conception of research ideas • design and implementation of models • running of large-scale experiments • writing and presentation of the paper	
Signature and Date		Sept. 29th 2022

Supervisor Confirmation

By signing the Statement of Authorship, the supervisor is certifying that the candidate made a substantial contribution to the publication, and that the description above is accurate.

Supervisor name	Prof. Andrew Zisserman	
Supervisor comments		
Signature and Date		Sept. 29th 2022

Statement of Authorship for the paper "Self-supervised Co-Training for Video Representation Learning" in Chapter 4.

Paper title	Self-supervised Co-Training for Video Representation Learning
Authors	**Tengda Han**, Weidi Xie, Andrew Zisserman
Publication status	Published
Publication details	Neural Information Processing Systems (NeurIPS), 2020.

Student Confirmation

Student name	Tengda Han	
Contribution to the paper	First-author contribution: • conception of research ideas • design and implementation of models • running of large-scale experiments • writing and presentation of the paper	
Signature and Date		Sept. 29th 2022

Supervisor Confirmation

By signing the Statement of Authorship, the supervisor is certifying that the candidate made a substantial contribution to the publication, and that the description above is accurate.

Supervisor name	Prof. Andrew Zisserman	
Supervisor comments		
Signature and Date		Sept. 29th 2022

Statement of Authorship for the paper "Prompting Visual-Language Models for Efficient Video Understanding" in Chapter 5.

Paper title	Prompting Visual-Language Models for Efficient Video Understanding
Authors	Chen Ju, **Tengda Han**, Kunhao Zheng, Ya Zhang, Weidi Xie
Publication status	Published
Publication details	European Conference on Computer Vision (ECCV), 2022.

Student Confirmation

Student name	Tengda Han	
Contribution to the paper	• conception and improvement of research ideas • pre-processing of video-language datasets • running of some experiments • writing of the paper	
Signature and Date		Sept. 29th 2022

Supervisor Confirmation

By signing the Statement of Authorship, the supervisor is certifying that the candidate made a substantial contribution to the publication, and that the description above is accurate.

Supervisor name	Prof. Andrew Zisserman	
Supervisor comments		
Signature and Date		Sept. 29th 2022

Statement of Authorship for the paper "Turbo Training with Token Dropout" in Chapter 6.

Paper title	Turbo Training with Token Dropout
Authors	**Tengda Han**, Weidi Xie, Andrew Zisserman
Publication status	Published
Publication details	British Machine Vision Conference (BMVC), 2022.

Student Confirmation

Student name	Tengda Han	
Contribution to the paper	First-author contribution: • conception of research ideas • design and implementation of models • running of large-scale experiments • writing and presentation of the paper	
Signature and Date		Sept. 29th 2022

Supervisor Confirmation

By signing the Statement of Authorship, the supervisor is certifying that the candidate made a substantial contribution to the publication, and that the description above is accurate.

Supervisor name	Prof. Andrew Zisserman	
Supervisor comments		
Signature and Date		Sept. 29th 2022

Statement of Authorship for the paper "Temporal Alignment Networks for Long-term Video" in Chapter 7.

Paper title	Temporal Alignment Networks for Long-term Video
Authors	**Tengda Han**, Weidi Xie, Andrew Zisserman
Publication status	Published
Publication details	[Oral] Conference on Computer Vision and Pattern Recognition (CVPR), 2022.

Student Confirmation

Student name	Tengda Han	
Contribution to the paper	First-author contribution: • conception of research ideas • design and implementation of models • downloading and pre-processing of large-scale video datasets • annotation of videos • running of large-scale experiments • writing and presentation of the paper	
Signature and Date		Sept. 29th 2022

Supervisor Confirmation

By signing the Statement of Authorship, the supervisor is certifying that the candidate made a substantial contribution to the publication, and that the description above is accurate.

Supervisor name	Prof. Andrew Zisserman	
Supervisor comments		
Signature and Date		Sept. 29th 2022

Statement of Authorship for the paper "AutoAD: Automatic Audio Description for Movies" in Chapter 8.

Paper title	AutoAD: Movie Description in Context
Authors	**Tengda Han**, Max Bain, Arsha Nagrani, Gül Varol, Weidi Xie, Andrew Zisserman
Publication status	Not published
Publication details	N/A.

Student Confirmation

Student name	Tengda Han	
Contribution to the paper	First-author contribution: • conception of research ideas • joint pre-processing of datasets • design and implementation of models • running of large-scale experiments • writing of the paper	
Signature and Date		Sept. 29th 2022

Supervisor Confirmation

By signing the Statement of Authorship, the supervisor is certifying that the candidate made a substantial contribution to the publication, and that the description above is accurate.

Supervisor name	Prof. Andrew Zisserman	
Supervisor comments		
Signature and Date		Sept. 29th 2022