Ph.D. DISSERTATION

Improving Efficiency in Large-Scale Self-Supervised Video Representation Learning

대규모 자기지도 비디오 표현학습법의 효율성 개선

FEBRUARY 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Sangho Lee

Improving Efficiency in Large-Scale Self-Supervised Video Representation Learning

대규모 자기지도 비디오 표현학습법의 효율성 개선

지도교수 김 건 희

이 논문을 공학박사학위논문으로 제출함

2022 년 11 월

서울대학교 대학원

컴퓨터 공학부

이 상 호

이 상 호의 공학박사 학위논문을 인준함

2022 년 12 월

위 원 장	유 승 주	(인)
부위원장	김 건 희	(인)
위 원	강 유	(인)
위 원	이 준 석	(인)
위 원	황 성 주	(인)

Abstract

Video is a very attractive data source for computer vision and machine learning; it contains dynamic and multimodal signals to learn from. Since adding annotations to videos is very expensive, self-supervised video representation learning has gained significant attention. However, self-supervised learning requires large-scale training, so we need large compute and memory resources. Furthermore, real-world videos are usually very noisy, so finding good video data to learn from requires human verification, which hinders large-scale data collection.

In this thesis, we explore these problems in self-supervised video representation learning and propose the following three solutions to improve learning efficiency. First, we investigate how to learn from unlabeled videos without decoding them. Videos are usually stored in a compressed format, e.g., MPEG, and decoding them requires significant compute resources. Our novel architecture and proposed pretext tasks allow us to learn from unlabeled compressed videos with minimal performance degradation and achieve fast video processing time. Second, we introduce a multimodal bidirectional Transformer architecture for self-supervised learning of contextualized audio-visual representation from unlabeled videos. End-to-end training of multimodal Transformers is challenging due to the large memory requirement of Transformer architecture. With our novel parameter reduction technique based on matrix decomposition with low-rank approximation, we successfully train our multimodal Transformer and achieve competitive results in various downstream tasks. Lastly, we propose an automatic and scalable data collection pipeline for self-supervised audio-

visual representation learning. We curate noisy video data using an MI-based subset selection algorithm. Audio and visual models trained on the resulting datasets yield competitive or better performance than those trained on existing, manually verified datasets. We release a large-scale open-domain video dataset, ACAV100M, consisting of 100M clips curated with our pipeline for audio-visual representation learning.

Keywords: Deep Learning, Computer Vision, Large-Scale Video Understanding, Self-Supervised Representation Learning

Student Number: 2017-26247

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the greatest goals of computer vision or machine learning is to learn perception in a way that humans and other biological agents can, which brings us to a question: how do we learn to perceive the world? If we look at babies to see how they learn to perceive, there are multiple essential components [6]. They learn from dynamic and multisensory signals. They also interact with their surroundings such as toys. The most characteristic part of early childhood learning is unsupervised. No one tells the babies which is input and which is output.

From this point of view, video is a very attractive data source for computer vision. It contains multimodal signals to learn from including visual frames, audio streams, and even sometimes language in the format of transcripts, as shown in Figure 1.1. We can also teach an agent to learn physical interactions such as object manipulation and state changes in simulated video environments [7, 8, 9].

The video understanding community has made many efforts so that we have superior video models that excel on a variety of tasks including action

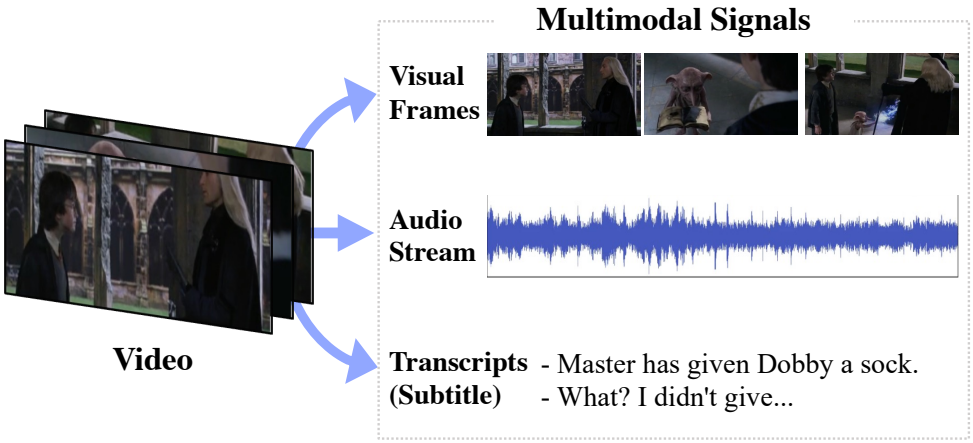


Figure 1.1: Videos provide abundant learning signals. We can learn expressive representations using multimodal supervision from visual frames, sound, and even sometimes speech transcripts.

recognition [10, 11, 12], video semantic segmentation [13, 14, 15, 16], video retrieval [17, 18, 19] and video QA [20, 21, 22, 23]. However, unlike the early childhood learning of humans, a lot of video understanding research has focused on fully-supervised learning to make a highly specialized video system optimized to be good at one task. If we give a video retrieval model a video segmentation problem, or vice versa, the model performs very poorly, which is not ideal for true video understanding. Furthermore, while training a video model from scratch in a supervised manner needs a lot of labels, adding annotations to videos is very expensive.

For this reason, learning from unlabeled videos using self-supervised methods has received significant attention [24, 25, 26, 27]. Figure 1.2 illustrates a popular scenario in learning deep self-supervised video representations. We design a *pretext task* that reflects the underlying nature of video data, but does not require manually annotated labels, and train our model from scratch us-

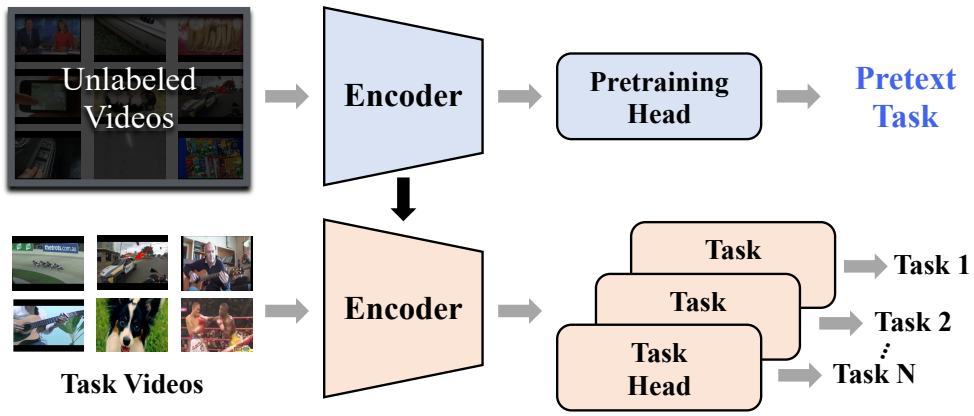


Figure 1.2: **Learning from unlabeled videos in a self-supervised manner**. We define a pretext task with artificially generated pseudo-labels from unlabeled videos and train our model from scratch using the task. Once we finish this pretraining step, we can transfer our model to different downstream tasks by attaching new prediction heads to different tasks and training the model on a target scenario.

ing the task. Once we finish this pretraining step, the learned representation can be transferred to different downstream tasks by attaching new prediction heads specific to different tasks. Here, self-supervised models are very good *few-shot* learners [28, 29]: the transferred model can achieve good downstream performance with relatively small finetuning data.

Now, the key question here is: how can we define the pretext task to perform well on downstream tasks? The time dimension in video data provides natural supervision signals, e.g., direction, ordering and speed, and we can leverage this temporal information to define the pretext task [30, 31, 32, 25, 33, 34]. We can also learn transformation-invariant representations [35, 36, 37]. However, all these methods utilize a single modality: visual frames. Since video is by nature

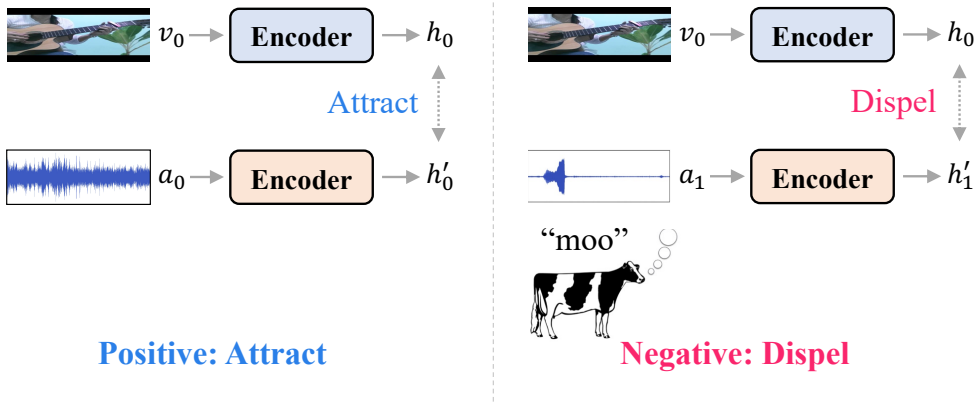


Figure 1.3: **Instance Discrimination**. We learn audio and visual representations in a contrastive manner: We minimize the distance between the features of audio and visual signals from the same video instance while maximizing the distance between the features of audio and visual signals from different video instances.

multimodal, as shown in Figure 1.1, there have been many attempts to learn cross-modal video representations, especially leveraging the natural correlation between audio and visual signals [38], and previous works have shown that these multimodal methods that use both audio and visual streams achieve better downstream performance than unimodal methods [39, 40].

Recently, to utilize the audio-visual correspondence, contrastive learning methods are often used [41, 42]. The goal of these methods is to discriminate between different instances [43]: By treating each video instance as a distinct class of its own, we enforce the features of audio and visual signals from the same video instance to be similar and contrast the features of audio & visual signals from different video instances (see Figure 1.3).

Problems that Hinder Efficiency in Self-Supervised Video Representation Learning To summarize, we can learn expressive video representations using audio-visual self-supervised methods without adding expensive annotations to videos. The learned representation needs relatively small data for finetuning on each downstream task. However, we need large-batch training because large-scale training is one key ingredient for the success of self-supervised learning [44, 45, 46]. Furthermore, we need large-scale video data for pretraining because self-supervised representations generally improve with more data [47, 48]. In this thesis, we explore the following three challenges incurred by this large-scale training on large-scale video data and propose novel solutions to them to improve learning efficiency.

Video decoding requires large compute resource. Training video models are notoriously resource-heavy, and part of it comes from the decoding step. Videos are usually stored in a compressed format, e.g., MPEG [49], and we need to decode frames before feeding them into traditional video models. The decoding step is very expensive, requiring about 10 to 20 times more preprocessing (decoding) time than inference time. Furthermore, it is mainly done on CPUs; this is very harmful to training deep neural networks because GPUs to load them on sit idle for most of the training time. These issues are exacerbated for self-supervised video representation learning because we need large-batch training. To get around this problem, several previous approaches proposed supervised approaches to operating directly on compressed videos [50, 51, 52, 53]. In this thesis, we extend this idea by learning from unlabeled compressed videos (Chapter 2).

Video Transformers require large memory footprint. In the field of natural language processing (NLP), Transformers have shown to be very powerful for learning contextualized embeddings [54, 55, 56, 57, 58, 46, 59, 60],

and several recent works have extended it to vision-and-language including VideoBERT [61], Oscar [62] and VIVO [63]. However, None of these models is end-to-end trained; they use a language-pretrained BERT [46] due to the large memory requirements of Transformers. Especially, this is more problematic for learning self-supervised video Transformers because Transformers scale quadratically with the length of the input sequence and each mini-batch of video instances for self-supervised learning usually contains thousands of frames. This thesis aims to train a multimodal Transformer in a self-supervised manner by leveraging audio-visual correlations. However, since we do not have pretrained visual/audio-pretrained BERTs, we need to train our model end-to-end. Thus, we need to reduce memory consumption during the model training. While there are several orthogonal approaches to reducing the memory requirements including parameter sharding [64], knowledge distillation [65] and network pruning [66], we propose a novel parameter sharing technique to reduce the model size (Chapter 3).

Finding "good" video data for audio-visual self-supervised learning in a scalable manner. Existing datasets for audio-visual learning contain at most eight months of video clips [5]. Compared to visual-text datasets, these are smaller by orders of magnitudes [67]. What makes this difference? While we can use Automatic Speech Recognition (ASR) to fully automatize visual-text data curation, all audio-visual data curation methods have relied on human workers to filter noisy correspondence between audio and visual channels [3, 5, 4]. To make matters worse, we cannot scrape random videos to increase the dataset size because some videos that we can easily see online, such as music videos, screencasts and news broadcasts, do not exhibit audio-visual correlation. This thesis aims to find a way to collect good video data for audio-visual self-supervised learning without human efforts, especially using an

automatic and scalable pipeline (Chapter 4).

1.1 Contributions

We summarize the main contributions of this thesis below.

- **Learning from Unlabeled Videos without Decoding (Chapter 2)**. Self-supervised learning of video representations has received great attention. Existing methods typically require frames to be *decoded* before being processed, which increases compute and storage requirements and ultimately hinders large-scale training. In this work, we propose an efficient self-supervised approach to learn video representations by eliminating the expensive decoding step. We use a three-stream video architecture that encodes I-frames and P-frames of a compressed video. Unlike existing approaches that encode I-frames and P-frames individually, we propose to *jointly* encode them by establishing bidirectional dynamic connections across streams. To enable self-supervised learning, we propose two pretext tasks that leverage the multimodal nature (RGB, motion vector, residuals) and the internal GOP structure of compressed videos. The first task asks our network to predict zeroth-order motion statistics in a spatio-temporal pyramid; the second task asks correspondence types between I-frames and P-frames after applying temporal transformations. We show that our approach achieves competitive performance on compressed video recognition both in supervised and self-supervised regimes.

  This work is published in:

  [68] Youngjae Yu*, **Sangho Lee***, Gunhee Kim, Yale Song. *Self-Supervised Learning of Compressed Video Representations*. **ICLR 2021**. (*: equal contribution)

- **Parameter Sharing Schemes for Multimodal Transformers (Chapter 3)**. The recent success of Transformers in the language domain has motivated adapting it to a multimodal setting, where a new visual model is trained in tandem with an already pretrained language model. However, due to the excessive memory requirements from Transformers, existing work typically fixes the language model and trains only the vision module, which limits its ability to learn cross-modal information in an end-to-end manner. In this work, we focus on reducing the parameters of multimodal Transformers in the context of audio-visual video representation learning. We alleviate the high memory requirement by sharing the parameters of Transformers across layers and modalities; we decompose the Transformer into modality-specific and modality-shared parts so that the model learns the dynamics of each modality both individually and together, and propose a novel parameter sharing scheme based on low-rank approximation. We show that our approach reduces parameters of the Transformers up to 97%, allowing us to train our model end-to-end from scratch. We also propose a negative sampling approach based on an instance similarity measured on the CNN embedding space that our model learns together with the Transformers. To demonstrate our approach, we pretrain our model on 30-second clips (480 frames) from Kinetics-700 and transfer it to audio-visual classification tasks.

  This work is published in:

  [69] **Sangho Lee**, Youngjae Yu, Gunhee Kim, Thomas Breuel, Jan Kautz, Yale Song. *Parameter Efficient Multimodal Transformers for Video Representation Learning.* **ICLR 2021**.

- **Massively Harvesting Good Video Data to Learn from with-**

out Human Efforts (Chapter 4). The natural association between visual observations and their corresponding sound provides powerful self-supervisory signals for learning video representations, which makes the ever-growing amount of online videos an attractive source of training data. However, large portions of online videos contain irrelevant audio-visual signals because of edited/overdubbed audio, and models trained on such uncurated videos have shown to learn suboptimal representations. Therefore, existing self-supervised approaches rely on datasets with predetermined taxonomies of semantic concepts, where there is a high chance of audio-visual correspondence. Unfortunately, constructing such datasets requires labor-intensive manual annotation and/or verification, which severely limits the utility of online videos for large-scale learning. In this work, we present an automatic dataset curation approach based on subset optimization where the objective is to maximize the mutual information between audio and visual channels in videos. We demonstrate that our approach finds videos with high audio-visual correspondence and show that self-supervised models trained on our data achieve competitive performances compared to models trained on existing manually curated datasets. The most significant benefit of our approach is scalability: We release ACAV100M that contains 100 million videos with high audio-visual correspondence, ideal for self-supervised video representation learning.

This work is published in:

[70] **Sangho Lee***, Jiwan Chung*, Youngjae Yu, Gunhee Kim, Thomas Breuel, Gal Chechik, Yale Song. *ACAV100M: Automatic Curation of Large-Scale Datasets for Audio-Visual Video Representation Learning.* **ICCV 2021**. (*: equal contribution)

1.2 Thesis Organization

This thesis is organized as follows. In Chapter 2-4, we introduce three independent studies that propose solutions to different challenging problems that hinder efficiency in self-supervised video representation learning: learning from unlabeled videos without decoding them (Chapter 2), parameter sharing schemes for multimodal Transformers (Chapter 3), and finding good video data to learn from without human efforts (Chapter 4). We conclude the thesis in Chapter 5.

Chapter 2

Learning from Unlabeled Videos without Decoding

2.1 Introduction

There has been significant progress on self-supervised learning of video representations. It learns from unlabeled videos by exploiting their underlying structures and statistics as free supervision signals, which allows us to leverage large amounts of videos available online. Unfortunately, training video models is notoriously difficult to scale. Typically, practitioners have to make trade-offs between *compute* (decode frames and store them as JPEG images for faster data loading, but at the cost of large storage) and *storage* (decode frames on-the-fly at the cost of high computational requirements). Therefore, large-batch training of video models is difficult without high-end compute clusters. Although these issues are generally applicable to any video-based scenarios, they are particularly problematic for self-supervised learning because large-scale training is one key ingredient [44, 45, 46] but that is exactly where these issues are aggravated.

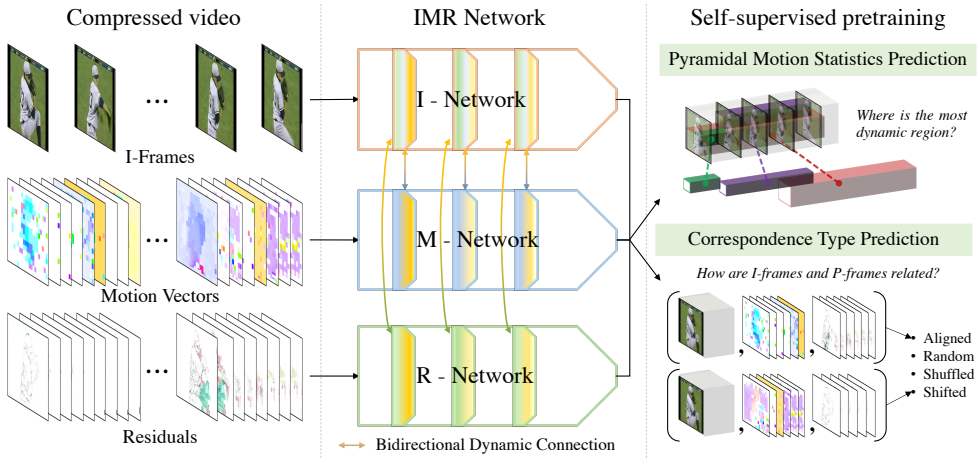


Figure 2.1: **IMR network** consists of three sub-networks encoding different information streams provided in compressed videos. We incorporate bidirectional dynamic connections to facilitate information sharing across streams. We train the model using two novel pretext tasks designed by exploiting the underlying structure of compressed videos.

Recently, several approaches demonstrated benefits of compressed video recognition [50, 51, 52, 53]. Without ever needing to decode frames, these approaches can alleviate compute and storage requirements, e.g., resulting in 3 to 10 times faster solutions than traditional video CNNs at a minimal loss on accuracy [51, 53]. Also, motion vectors embedded in compressed videos provide a free alternative to optical flow which is compute-intensive; leveraging this has been shown to be two orders of magnitude faster than optical flow-based approaches [52]. However, all the previous work on compressed video has focused on supervised learning and there has been no study that shows the potential of compressed videos in self-supervised learning; this is the focus of our work.

In this work, we propose a self-supervised approach to learning video representations directly in the compressed video format. We exploit two inherent

characteristics of compressed videos: First, video compression packs a sequence of images into several Group of Pictures (GOP). Intuitively, the GOP structure provides *atomic representation of motion*; each GOP contains images with just enough scene changes so a video codec can compress them with minimal information loss. Because of this atomic property, we enjoy less spurious, more consistent motion information at the GOP-level than at the frame-level. Second, compressed videos naturally provide multimodal representation (*i.e.* RGB frames, motion vectors, and residuals) that we can leverage for multimodal correspondence learning. Based on these, we propose two novel pretext task (see Figure 2.1): The first task asks our model to predict zeroth-order motion statistics (*e.g.* where is the most dynamic region) in a pyramidal spatio-temporal grid structure. The second involves predicting correspondence types between I-frames and P-frames after temporal transformation. Solving our tasks require *implicitly* locating the most salient moving objects and matching their appearance-motion correspondences between I-frames and P-frames; this encourages our model to learn discriminative representation of compressed videos.

A compressed video contains three streams of multimodal information – *i.e.* RGB images, motion vectors, and residuals – with a dependency structure between an I-frame stream and the two P-frame streams punctuated by GOP boundaries. We design our architecture to encode this dependency structure; it contains one CNN encoding I-frames and two other CNNs encoding motion vectors and residuals in P-frames, respectively. Unlike existing approaches that encode I-frames and P-frames individually, we propose to *jointly* encode them to fully exploit the underlying structure of compressed videos. To this end, we use a three-stream CNN architecture and establish *bidirectional dynamic connections* going from each of the two P-frame streams into the I-frame stream, and vice versa, and put these connections layer-wise to learn the correlations between

them at multiple spatial/temporal scales (see Figure 2.1). These connections allow our model to fully leverage the internal GOP structure of compressed videos and effectively capture atomic representation of motion.

In summary, our main contributions are two-fold: (1) We propose a three-stream architecture for compressed videos with bidirectional dynamic connections to fully exploit the internal structure of compressed videos. (2) We propose novel pretext tasks to learn from compressed videos in a self-supervised manner. We demonstrate our approach by pretraining the model on Kinetics-400 [71] and finetuning it on UCF101 [1], HMDB51 [72]. Our model achieves new state-of-the-art performance in compressed video classification tasks in both supervised and self-supervised regimes, while maintaining a similar computational efficiency as existing compressed video recognition approaches [51, 52].

2.2 Approach

We use videos compressed according to the MPEG-4 Part 2 specifications [49] as our input, following the previous work [51, 52, 53]. This compression format encodes an RGB image sequence as a series of GOPs (Group of Pictures) where each GOP starts with one I-frame followed by a variable number of P-frames. An I-frame stores RGB values of a complete image and can be decoded on its own. A P-frame holds only the changes from the previous reference frame using motion vectors and residuals. The motion vectors store 2D displacements of the most similar patches between the reference and the target frames, and the residuals store pixel-wise differences to correct motion compensation errors. We use all the three modalities contained in compressed videos as our input.

Formally, our input is T GOPs, $G_0, \cdots, G_{T-1}$, where each G_t contains one I-frame $I_t \in \mathbb{R}^{H \times W \times 3}$ followed by $K-1$ pairs of motion vectors $M_{t,k} \in \mathbb{R}^{H \times W \times 2}$

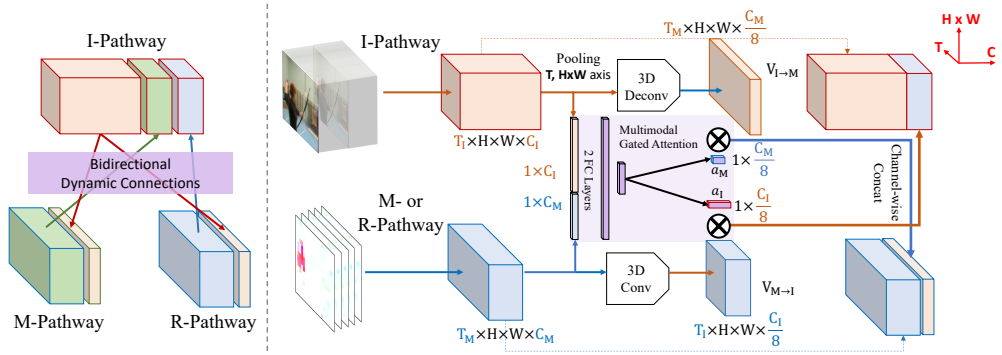


Figure 2.2: **Bidirectional dynamic connections** are established between I & M pathways and between I & R pathways to facilitate information sharing between I-frames and P-frames. We incorporate *multimodal-gated attention* to dynamically modulate the connections based on input. Feature tensors (orange and blue cubes) are placed in the T-HxW-C plane.

and residuals $R_{t,k} \in \mathbb{R}^{H \times W \times 3}$, $k \in [1, K)$. For efficiency and simplicity, we assume an identical GOP size K for all $t \in [0, T)$.

2.2.1 IMR Network for Compressed Videos

Our model consists of three CNNs, each with 3D convolutional kernels modeling spatio-temporal dynamics within each input stream $\{I_t\}$, $\{M_{t,k}\}$, $\{R_{t,k}\}$, $t \in [0, T), k \in [0, K)$; we denote these sub-networks by I-network f_I, M-network f_M, and R-network f_R, respectively, and call our model *IMR Network (IMRNet)*. We account for the difference in the amount of information between I-frames and P-frames by adjusting the capacity of networks accordingly. Specifically, following Wu et al. [51], we make the capacity of f_I larger than f_M and f_R by setting the number of channels in each layer of f_I to be γ times higher than those of f_M and f_R (we set $\gamma = 16$).

Existing models for compressed videos typically perform late fusion [51, 52],

i.e., they combine embeddings of I-frames and P-frames only *after encoding* each stream. However, we find that it is critical to allow our sub-networks to share information *as they encode* their respective input streams. To this end, we establish layer-wise lateral connections between f_I & f_M and between f_I & f_R.

Bidirectional Dynamic Connections. Lateral connections have been used to combine information from different streams, e.g., RGB images and optical flow images [73], and RGB images sampled at different frame rates [11]. In this work, we use it to combine information from I-frames and P-frames. Our approach is different from previous work in two key aspects: (1) We establish *bidirectional* connections between streams, instead of unidirectional connections as was typically done in the past [73, 11], so that information sharing is symmetrical between streams. (2) We incorporate *multimodal gated attention* to dynamically adjust the connections based on multimodal (I-frame and P-frames) information. We call our approach *bidirectional dynamic connections* to highlight these two aspects and differentiate ours from previous work, e.g., SlowFast networks [11] establish *unidirectional* lateral connections and the connections are *static* regardless of the content from the other stream.

We combine embeddings from different sub-networks via channel-wise concatenation, which requires embeddings to match their spatio-temporal dimensions. However, f_I processes κ times less frames than f_M and f_R, producing embeddings that are κ times smaller in the temporal dimension. Therefore, we transform the embeddings with time-strided 3D (de-)convolution with $(\kappa \times 1 \times 1)$ kernels, $C/8$ channels, and $(\kappa, 1, 1)$ temporal stride: We use convolution for $f_M/f_R \rightarrow f_I$ to decrease the time dimension and deconvolution for $f_I \rightarrow f_M/f_R$ to increase it. Note that simply using the (de-)conv layers will perform *static* transformation regardless of what is provided from the other sub-network, similar to Feichtenhofer et al. [11]. However, we find it critical to

make the transformations aware of information from both sub-networks so that the networks can dynamically adjust the connections and selectively share only the most relevant information from each sub-network.

To achieve this, we dynamically modulate (de-)conv layer outputs using multimodal-gated attention weights. Let $\mathbf{x}_I \in \mathbb{R}^{T_I \times W \times H \times C_I}$ and $\mathbf{x}_M \in \mathbb{R}^{T_M \times W \times H \times C_M}$ be the embeddings from f_I and f_M, respectively. We max-pool $\mathbf{x}_I$ and $\mathbf{x}_M$ and concatenate them to obtain multimodal embedding $\mathbf{z} \in \mathbb{R}^{C_Z}$ with $C_Z = C_I + C_M$. We define multimodal gate functions that take as input $\mathbf{z}$ and generate attention weights $\mathbf{a}_I \in \mathbb{R}^{C_I/8}$ and $\mathbf{a}_M \in \mathbb{R}^{C_M/8}$ as

$$\mathbf{a}_I = \sigma\left(W_3\mathbf{h} + b_3\right),\ \mathbf{a}_M = \sigma\left(W_4\mathbf{h} + b_4\right),\ \mathbf{h} = \zeta\left(W_2\zeta\left(W_1\mathbf{z} + b1\right) + b2\right) \quad (2.1)$$

where σ is a sigmoid function, ζ is a Leaky ReLU function, and $W_1, W_2 \in \mathbb{R}^{C_Z \times C_Z}, b_1, b_2 \in \mathbb{R}^{C_Z}, W_3 \in \mathbb{R}^{C_I/8 \times C_Z}, b_3 \in \mathbb{R}^{C_I/8}, W_4 \in \mathbb{R}^{C_M/8 \times C_Z}, b_4 \in \mathbb{R}^{C_M/8}$ are weight parameters. Next, we use these attention weights to modulate the (de-)conv output embeddings,

$$\mathbf{v}_{M\to I} = \mathbf{a}_I \otimes \texttt{3d\_conv}(\mathbf{x}_M),\ \mathbf{v}_{I\to M} = \mathbf{a}_M \otimes \texttt{3d\_deconv}(\mathbf{x}_I) \quad (2.2)$$

where $\otimes$ is channel-wise multiplication. We repeat the same process for f_I & f_R to obtain $\mathbf{v}_{R\to I}$ and $\mathbf{v}_{I\to R}$, and combine them with the feature embeddings via channel-wise concatenation,

$$\hat{\mathbf{x}}_I = [\mathbf{x}_I; \mathbf{v}_{M\to I}; \mathbf{v}_{R\to I}],\ \hat{\mathbf{x}}_M = [\mathbf{x}_M; \mathbf{v}_{I\to M}],\ \hat{\mathbf{x}}_R = [\mathbf{x}_R; \mathbf{v}_{I\to R}] \quad (2.3)$$

Each of these is fed into the next layer in the corresponding sub-network. We establish these lateral connections across multiple layers of our network. To obtain the final embedding, we apply average pooling on the output from the final layer of each sub-network and concatenate them channel-wise.

Note that the design of IMRNet is orthogonal to the design of video CNNs; while we adapt 3D ResNet [74] as the backbone in our experiments, we can use

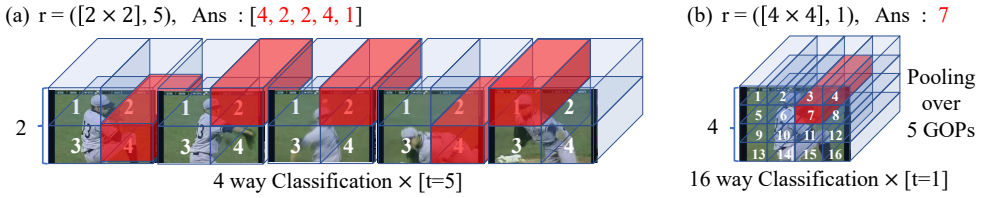


Figure 2.3: **Pyramidal motion statistics prediction** asks our network to find a region with the highest energy of motion. Here we visualize two levels in a spatio-temporal pyramid for illustration.

any of existing CNN architectures as the backbone, e.g., C3D [75], I3D [76], R(2+1)D [77]. What is essential, however, is that (i) there are three sub-networks, each modeling one of the three input streams, and (ii) information from different networks are combined via bidirectional dynamic connections as they are encoded.

2.2.2 Self-Supervised Learning Objectives

Compressed videos have unique properties, i.e., the multimodal nature of information (RGB, motion vector, residuals) and the internal GOP structure that provides atomic representation of motion. We turn these properties into free self-supervisory signals and design two novel pretext tasks.

Pyramidal Motion Statistics Prediction (PMSP). One important desideratum of video CNNs is learning visual representation that captures salient objects and motion. We hypothesize that there is an implicit *videographer bias* captured in videos in-the-wild that naturally reflect visual saliency: Videos are purposely recorded to highlight important objects and their movements.[1] Therefore, regions with the highest energy of motion can provide clues

[1]This is, of course, a weak hypothesis. But we show some convincing empirical evidence in Section 2.3.3.

to learning the desired video representation. We can easily find those regions in compressed videos: the motion vectors in P-frames readily provide magnitude and angular information of motion, which we can harness to find the most vibrant regions.

Based on this intuition, we design a task that asks our model to predict the zeroth-order motion statistics (i.e., the most vibrant region) in a given video. For this, we must be able to deal with a variety of object sizes because a salient moving object can appear at any location in any size. A classical solution to this is to perform pyramidal prediction [78, 79]: We divide a video into spatio-temporal 3D grids at multiple scales and ask our network to predict the most vibrant region at each scale.

Specifically, we define a pyramidal classification task with the following loss function,

$$\mathcal{L}_{PMSP} = -\sum_{i}\sum_{r}\sum_{q} y_{q,r}^{(i)} \cdot \log \alpha_r \left(\mathbf{x}_{q,r}^{(i)}\right) \tag{2.4}$$

This is a cross-entropy loss computed at every q-th grid in every r-th level of a spatio-temporal pyramid; i is the sample index. We define a 9-level spatio-temporal pyramid with 3 spatial and 3 temporal scales, i.e., $r \in \{(s,t)|s \in \{[2\times 2],[3\times 3],[4\times 4]\}, t \in \{1,3,5\}\}$. The index q iterates over all possible temporal coordinates in the r-th level of the pyramid, e.g., in Figure 2.3 (a), $q \in [0,\cdots,4]$ with $r = ([2\times 2], 5)$. $y_{q,r}^{(i)}$ is a one-hot label marking the location with the highest energy of motion in the q-th grid in r-th level in the pyramid, e.g., in Figure 2.3 (a), $y_{q,r}^{(i)}$ is a 4-dimensional one-hot vector. $\mathbf{x}_{q,r}^{(i)}$ is the (q,r)-th feature in a 3D grid; we concatenate output embeddings from all three sub-networks, $\mathbf{x}^{(i)} = [\mathbf{x}_I^{(i)}; \mathbf{x}_M^{(i)}; \mathbf{x}_R^{(i)}]$. Finally, $\alpha_r(\cdot)$ is a 2-layer MLP with a softmax classifier predicting the most vibrant region in the given grid; we define one such classifier for each r.

Algorithm 1: Self-Supervision Labels for Pyramidal Motion Statistics Prediction

Require: T GOPs each of which has $K-1$ motion vectors $\{M_{0,1}, ..., M_{T-1,K-1}\}$

Generate dx, dy by convolving motion vectors with Prewitt operator [80], G_x, G_y
$t \leftarrow 1$ ▷ Set temporal scale t to 1
$Y \leftarrow []$ ▷ Empty list for labels
for $i = 0$ **to** 2 **do**
  for $n = 0$ **to** $t - 1$ **do**
    $sum\_dx \leftarrow \text{sum}(dx[n * K : (n + T - t + 1) * K])$
    $sum\_dy \leftarrow \text{sum}(dy[n * K : (n + T - t + 1) * K])$
    $magnitude \leftarrow \text{cartToPolar}(sum\_dx, sum\_dy)$
    $magnitude_{[2\times2]} \leftarrow \text{makeGrid}(magnitude, spatial = 2)$
    $magnitude_{[3\times3]} \leftarrow \text{makeGrid}(magnitude, spatial = 3)$
    $magnitude_{[4\times4]} \leftarrow \text{makeGrid}(magnitude, spatial = 4)$
    $y_{[2\times2]} \leftarrow \text{argmax}_{q\in[1,\cdots,2^2]}(magnitude_{[2\times2]})$
    $y_{[3\times3]} \leftarrow \text{argmax}_{q\in[1,\cdots,3^2]}(magnitude_{[3\times3]})$
    $y_{[4\times4]} \leftarrow \text{argmax}_{q\in[1,\cdots,4^2]}(magnitude_{[4\times4]})$
    $Y \leftarrow Y \cup [y_{[2\times2]}, y_{[3\times3]}, y_{[4\times4]}]$
  end for
  $t \leftarrow t + 2$
end for
return PMSP labels, Y, at multiple scales $\{(s,t)|s \in \{[2\times2], [3\times3], [4\times4]\}, t \in \{1,3,5\}\}$

Algorithm 1 shows a pseudo-code to compute the labels at multiple spatio-temporal scales, $r \in \{(s,t)|s \in \{[2\times2], [3\times3], [4\times4]\}, t \in \{1,3,5\}\}$.

Correspondence Type Prediction (CTP). One idea often used in self-supervision is applying certain transformations to data and asking a network to predict the correspondence type given a pair of instances (e.g., true pair or randomly selected pair) [39, 81, 82, 83]. The multimodal nature of compressed videos makes them an ideal data format to apply such self-supervision technique: The three frame types in compressed videos exhibit different characteristics, yet they are strongly correlated with each other. This allows us to consider

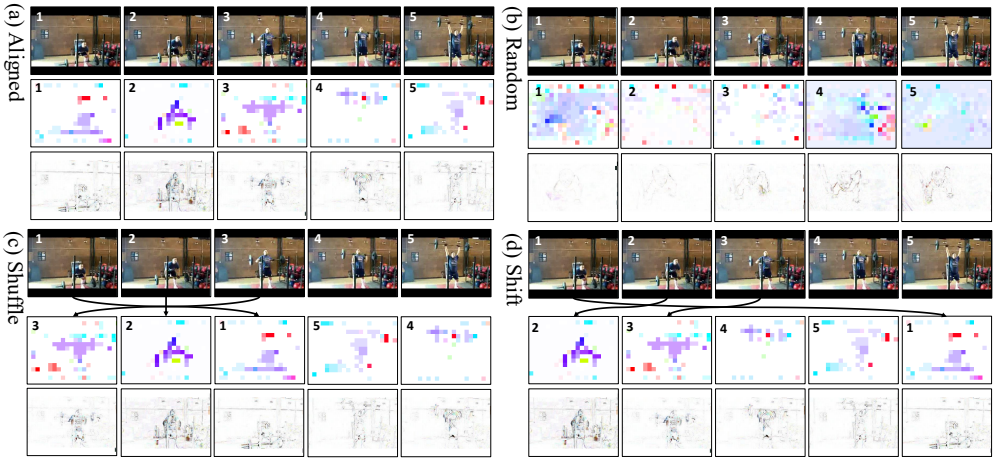


Figure 2.4: **Correspondence type prediction** asks our network to categorize different types of data transformations applied on P-frames. We illustrate four transformation types used in our experiments.

I-frames as a heavily transformed version of the corresponding P-frames, and vice versa. Learning the correspondence type between I-frames and P-frames can therefore encourage our network to learn discriminative representation of videos.

We define a correspondence type prediction task with the following loss function,

$$\mathcal{L}_{CTP} = -\sum_{i}\sum_{j} y_j^{(i)} \cdot \log \beta \left(\mathbf{x}_I^{(i)}, \mathcal{T}(\mathbf{x}_M^{(i)}, \mathbf{x}_R^{(i)}, j) \right) \tag{2.5}$$

where i is the sample index and j iterates over a set of transformations. $y_j^{(i)}$ is a one-hot label indicating different correspondence types determined by the type of transformation done, and $\mathcal{T}(\cdot, j)$ is a data transformation function that changes the input using the j-th transformation. We define four transformation types (see Figure 2.4): (1) **Aligned** keeps the original input (no transformation), (2) **Random** replaces the data with P-frames from a randomly selected video, (3) **Shuffle** randomly shuffles the GOP order, (4) **Shift** randomly divides GOPs

into two groups and switch the order, e.g., $[1, 2, 3, 4, 5]$ to $[2, 3, 4, 5, 1]$. Finally, $\beta(\cdot)$ is a 2-layer MLP with a softmax classifier.

Note that there is a nuanced difference between random P-frames and shuffled/shifted P-frames. The former contains P-frames that come from a different clip, while the latter contains P-frames of the same clip as the I-frames, yet in a different frame order. Intuitively, the former encourages our network to learn from global (clip-level) correspondence, while the latter formulates a local (frame-level) correspondence task. Therefore, our CTP task encourages our network to learn discriminative representations at both global and local levels. We provide empirical evidence showing the importance of this global-local mixed objective in Section 2.3.2.

Final Objective. We optimize our model using a learning objective $\mathcal{L}_{PMSP} + \lambda\mathcal{L}_{CTP}$ with $\lambda = 1$. The classifiers α_r and β are used only during self-supervised training; we detach them thereafter.

2.3 Experiments

Architecture Details. Table 2.1 provides architecture details of our IMRNet. In our experiments, we use both 3D ResNet-18 and 3D ResNet-50 as the backbone; we provide the details of both models in the table. We also include the details of our bidirectional dynamic connections, which include 3D convolutional/deconvolutional layers that downsamples/upsamples the computed features along the temporal dimension. We establish the connections after {*conv1, res2, res3, res4*} layers, each with different numbers of channels.

Optimization. We pretrain our model end-to-end from scratch for 20 epochs, including the initial warm-up period of 5 epochs. For downstream scenarios, we finetune our model for 500 epochs for UCF101 and for 300 epochs for HMDB51, including the warm-up period of 30 epochs. For both the pretraining and fine-

Stage	I Pathway	M/R Pathway	Output sizes $T \times S^2$
raw clip	–	–	60×224^2
data layer	stride $12, 1^2$	stride $2, 1^2$	I: 5×224^2 M/R: 25×224^2
$conv_1$	$1 \times 7^2, 64$ stride $1, 2^2$	$5 \times 7^2, 8$ stride $1, 2^2$	I: 5×112^2 M/R: 25×112^2
$pool_1$	$1 \times 3^2, max$ stride $1, 2^2$	$5 \times 3^2, max$ stride $1, 2^2$	I: 5×56^2 M/R: 25×56^2
res_2	(3D ResNet-18) $\begin{bmatrix} 1 \times 3^2, 64 \\ 1 \times 3^2, 64 \end{bmatrix} \times 2$ (3D ResNet-50) $\begin{bmatrix} 1 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	(3D ResNet-18) $\begin{bmatrix} 3 \times 3^2, 4 \\ 1 \times 3^2, 4 \end{bmatrix} \times 2$ (3D ResNet-50) $\begin{bmatrix} 3 \times 1^2, 4 \\ 1 \times 3^2, 4 \\ 1 \times 1^2, 16 \end{bmatrix} \times 3$	I: 5×56^2 M/R: 25×56^2
res_3	(3D ResNet-18) $\begin{bmatrix} 1 \times 3^2, 128 \\ 1 \times 3^2, 128 \end{bmatrix} \times 2$ (3D ResNet-50) $\begin{bmatrix} 1 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{bmatrix} \times 4$	(3D ResNet-18) $\begin{bmatrix} 3 \times 3^2, 8 \\ 1 \times 3^2, 8 \end{bmatrix} \times 2$ (3D ResNet-50) $\begin{bmatrix} 3 \times 1^2, 8 \\ 1 \times 3^2, 8 \\ 1 \times 1^2, 32 \end{bmatrix} \times 4$	I: 5×28^2 M/R: 25×28^2
res_4	(3D ResNet-18) $\begin{bmatrix} 3 \times 3^2, 256 \\ 1 \times 3^2, 256 \end{bmatrix} \times 2$ (3D ResNet-50) $\begin{bmatrix} 3 \times 1^2, 256 \\ 1 \times 3^2, 256 \\ 1 \times 1^2, 1024 \end{bmatrix} \times 6$	(3D ResNet-18) $\begin{bmatrix} 3 \times 3^2, 16 \\ 1 \times 3^2, 16 \end{bmatrix} \times 2$ (3D ResNet-50) $\begin{bmatrix} 3 \times 1^2, 16 \\ 1 \times 3^2, 16 \\ 1 \times 1^2, 64 \end{bmatrix} \times 6$	I: 5×14^2 M/R: 25×14^2
res_5	(3D ResNet-18) $\begin{bmatrix} 3 \times 3^2, 512 \\ 1 \times 3^2, 512 \end{bmatrix} \times 2$ (3D ResNet-50) $\begin{bmatrix} 3 \times 1^2, 512 \\ 1 \times 3^2, 512 \\ 1 \times 1^2, 2048 \end{bmatrix} \times 3$	(3D ResNet-18) $\begin{bmatrix} 3 \times 3^2, 32 \\ 1 \times 3^2, 32 \end{bmatrix} \times 2$ (3D ResNet-50) $\begin{bmatrix} 3 \times 1^2, 32 \\ 1 \times 3^2, 32 \\ 1 \times 1^2, 128 \end{bmatrix} \times 3$	I: 5×7^2 M/R: 25×7^2

Stage	$conv_1$	res_2	res_3	res_4
I to M/R	$1 \times 5^2, 8$ stride $5, 1^2$	$1 \times 5^2, 8$ stride $5, 1^2$	$1 \times 5^2, 16$ stride $5, 1^2$	$1 \times 5^2, 32$ stride $5, 1^2$
M/R to I	$5 \times 7^2, 4$ stride $5, 1^2$	$5 \times 7^2, 4$ stride $5, 1^2$	$5 \times 7^2, 8$ stride $5, 1^2$	$5 \times 7^2, 16$ stride $5, 1^2$

Table 2.1: **IMRNet architecture details**. We show two versions of IMRNet with different backbones: 3D ResNet-18 and 3D ResNet-50. We denote the input dimensions by {*temporal size, spatial size*^{2}}, kernels by {*temporal size, spatial size*^{2}*, channel size*} and strides by {*temporal stride, spatial stride*^{2}}.

tuning stages, we use SGD with momentum 0.9, weight decay 10^{-4}, and half-period cosine learning rate schedule. We use 4 NVIDIA Tesla V100 GPUs and use a batch size of 100.

Data. We pretrain our model on Kinetics-400 [71]. For evaluation, we finetune the pretrained model for action recognition using UCF101 [1] and HMDB51 [72]. We use 2-second video clips encoded in 30 FPS with a GOP size $T = 12$. We use all $T = 5$ GOPs but subsample every other P-frames within each GOP; this results in 5 I-frames and 25 P-frames. We randomly crop 224×224 pixels from videos resized to 256 pixels in the shorter side while keeping the aspect ratio. For data augmentation, we resize the video with various scales [.975, .9, .85] and apply random horizontal flip. For test videos, we take three equidistant 224×224 pixel crops from videos resized to 256 pixels to fully cover the spatial region. We approximate the fully-convolutional testing [10] by averaging the softmax scores for final prediction.

2.3.1 Supervised Learning Experiments

We first demonstrate our proposed IMR network in the fully-supervised setup, training it without using our self-supervised pretext tasks. We use the standard training and evaluation protocols for both UCF101 and HMDB51. For fair comparisons with existing approaches [51, 52], we report results both when we train the model from scratch and when we pretrain it on Kinetics-400 and finetune it on downstream datasets (indicated in column `Pretrain`).

Table 2.2 summarizes the results. When trained from scratch, our model outperforms CoViAR [51] by a large margin regardless of the chosen backbone. The performance gap is alleviated when the models are pretrained on Kinetics400, but our approach continues to outperform them even in this scenario. This suggest that CoViAR struggles to learn discriminative representations

Models	OF	Pretrain	Backbone	UCF	HMDB
CoViAR‡	✗	Scratch	ResNet-152	43.8	27.3
IMR (No connection)	✗	Scratch	3D ResNet-18	52.7	34.6
IMR (Unidirectional)	✗	Scratch	3D ResNet-18	69.7	40.8
IMR (No conv)	✗	Scratch	3D ResNet-18	71.7	42.6
IMR (No attention)	✗	Scratch	3D ResNet-18	73.2	43.5
IMRNet	✗	Scratch	3D ResNet-18	74.1	43.7
IMRNet	✗	Scratch	3D ResNet-50	80.2	55.9
CoViAR†	✗	ImageNet	ResNet-152 (I), ResNet-18 (P)	90.4	59.1
CoViAR‡	✗	K400	ResNet-152	90.8	59.2
IMRNet (Ours)	✗	K400	3D ResNet-18	91.4	62.8
IMRNet (Ours)	✗	K400	3D ResNet-50	**92.6**	**67.8**
CoViAR†	✓	ImageNet	ResNet-152 (I), ResNet-18 (P, OF)	94.9	70.2
DMC-Net†	✓	ImageNet	ResNet-152 (I), ResNet-18 (P)	90.9	62.8
DMC-Net†	✓	ImageNet	ResNet-152 (I), I3D (P)	92.3	71.8
IMRNet (Ours)	✓	K400	3D ResNet-50 (I, P), I3D (OF)	**95.1**	**72.2**

Table 2.2: **Results from the supervised setting**. Column `OF` indicates results using optical flow during training. Column `Pretrain` indicates datasets used for supervised pretraining. †: published results. ‡: our results based on official implementations by the authors. **Datasets**. K: Kinetics, UCF: UCF101, HMDB: HMDB51.

without help from a large-scale pretraining data. We believe the performance gap comes from the difference in how the two models encode compressed videos: CoViAR combines information from I-frames and P-frames only after encoding them separately, while we combine them in the early layers of CNN.

CoViAR and DMC-Net [52] reported improved results when they are trained using optical flow. Therefore, we also conduct experiments by adding an I3D network [76] to encode optical flow images; we simply concatenate our IMRNet features with the I3D features as our final representation (no lateral connections between IMRNet and I3D). This model outperforms both CoViAR and DMC-Net trained with optical flow (bottom group, Table 2.2). DMC-Net improves upon CoViAR by adapting GANs [84] to reconstruct optical flow from P-frames. Note that our approach (with 3D ResNet-50 backbone) outperforms DMC-Net (with ResNet-152/18 backbones) on both datasets *even without using optical flow* during training and thus significantly simplifies the training setup (no GANs required).

Models	R152*	R(2+1)D†	CoViAR‡	DMC‡	IMR‡ (3R18)	IMR‡ (3R50)
Preprocess (ms)	75.00	75.00	2.87	2.87	2.87	2.87
Inference (ms)	7.50	1.74	1.30	1.91	1.36	1.44
Total (ms)	82.50	76.74	4.17	4.78	4.23	4.31
GFLOPs	11.3	0.96	4.2	4.4	0.66	1.04

Table 2.3: **Runtime analysis** of per-frame speed (ms) and FLOPs. The number of input frames are different across models: * 1 frame (since it is a 2D CNN), † 16 frames, ‡ 25 frames. **Models**. R152: ResNet-152, DMC: DMC-Net, 3R18: 3D ResNet-18, 3R50: 3D ResNet-50.

Next, we conduct an ablation study on the bidirectional dynamic connection: (a) `No connection` removes lateral connections and thus is similar to CoViAR, (b) `Unidirectional` establishes connections from M/R-Networks to I-Network, but not vice versa, i.e., Equation 2.3 becomes $\hat{\mathbf{x}}_M = \mathbf{x}_M$, $\hat{\mathbf{x}}_R = \mathbf{x}_R$, (c) `No conv` replaces (de-)conv layers with simple up/down-sampling, (d) `No attention` removes the multimodal-gated attention module. The results are shown in Table 2.2. We can see that lateral connections are critical component of our model (`Ours` vs. `No connection`) and doing so in a bidirectional fashion significantly improves performance (`Ours` vs. `Unidirection`). We can also see that using (de-)conv layers and dynamically modulating the connection with gate functions improve performance (`Ours` vs. `No conv` and `No attention`).

Table 2.3 shows *per-frame* runtime speed (ms) and GFLOPs measured on an NVIDIA Tesla P100 GPU with Intel E5-2698 v4 CPUs ($*$ process individual frames. † and ‡ process 16- and 25-frame sequences, respectively). Our approach has the same preprocessing time of CoViAR and DMC-Net because all three approaches use the same video loader implementation [51]. As for the inference speed, IMRNet is comparable to CoViAR and even slightly faster than DMC-Net (we divide the total inference time by #frames following the convention of Wu et al. [51]). This is partly because we use lighter backbones (R18/R50 vs. R152 used in CoViAR and DMC-Net) to compensate for the expensive 3D convolutional operations, while DMC-Net requires an OF generator network of

7 all-convolutional layers, which adds extra cost. In terms of per-frame FLOPs, ours is more efficient than CoViAR and DMC-Net because the computation is done at the sequence-level rather than per-frame; we observe a similar trend for R(2+1)D (which uses ResNet-18) vs. ResNet-152. This shows that our 3D CNN backbones do not bring any significant extra cost compared to CoViAR and DMC-Net, and thus our model enjoys all the computational benefits of compressed video processing.

2.3.2 Self-supervised Learning Experiments

We move to the self-supervised regime and demonstrate our pretext tasks by pretraining our IMRNet on Kinetics-400 and transferring it to action recognition. Because ours is the first self-supervised approach to learn compressed video representation, there exist no published baseline that we can directly compare with. Therefore, we provide results from existing self-supervised approaches that require the decoding step. We include approaches that learn from RGB images – **AOT** [34], **Rotation** [37], **MotPred** [85], **RotNet3D** [37], **ST-Puzzle** [35], **ClipOrder** [86], **DPC** [87] – as well as those that learn from audio and visual channels in videos – **Multisensory** [39], **AVTS** [41], **ELo** [40].

Table 2.4 summarizes the results. We first notice that pretraining the models with any pretext tasks improves downstream performance (the first group of results), suggesting self-supervised pretraining is effective in general. We also see that IMRNet pretrained using our pretext tasks (PMSP+CTP) outperforms the baseline pretext tasks (second group) and self-supervised methods for uncompressed videos (third group). This shows the effectiveness of our IMRNet pretrained with our pretext tasks.

Next, we conduct an ablation study by pretraining the base models using either PMSP and CTP alone. We also test `CTP (Binary)` which is a simplied

Models	Compressed	Modality	Pretext	Pretrain	Backbone	UCF	HMDB
C3D	✗	**V**	MotPred	K400	C3D	61.2	33.4
3D ResNet-18	✗	**V**	RotNet3D	K600	3D ResNet-18	62.9	33.7
3D ResNet-18	✗	**V**	ST-Puzzle	K400	3D ResNet-18	65.8	33.7
R(2+1)D-18	✗	**V**	ClipOrder	UCF	R(2+1)D-18	72.4	30.9
3D ResNet-34	✗	**V**	DPC	K400	3D ResNet-34	75.7	35.7
Multisensory	✗	**A+V**	Multisensory	K400	Audio-VisualNet	82.1	–
AVTS	✗	**A+V**	AVTS	AS	MC3	89.0	61.6
ELo	✗	**A+V**	ELo	K400	(2+1)D ResNet-50	93.8	67.4
CoViAR‡	✓	**V**	Scratch	None	ResNet-152	43.8	27.3
IMRNet	✓	**V**	Scratch	None	3D ResNet-18	74.1	43.7
CoViAR‡	✓	**V**	AOT	K400	ResNet-152	53.6	29.3
CoViAR‡	✓	**V**	Rotation	K400	ResNet-152	56.7	31.4
IMRNet	✓	**V**	InfoNCE	K400	3D ResNet-18	73.9	43.7
IMRNet	✓	**V**	AOT	K400	3D ResNet-18	74.6	44.0
IMRNet	✓	**V**	Rotation	K400	3D ResNet-18	75.1	44.3
CoViAR‡	✓	**V**	PMSP	K400	ResNet-152	63.5	35.9
CoViAR‡	✓	**V**	CTP	K400	ResNet-152	64.4	37.4
CoViAR‡	✓	**V**	CTP (Binary)	K400	ResNet-152	63.7	37.1
IMRNet	✓	**V**	PMSP	K400	3D ResNet-18	76.0	44.9
IMRNet	✓	**V**	CTP	K400	3D ResNet-18	76.7	44.8
IMRNet	✓	**V**	CTP (Binary)	K400	3D ResNet-18	74.6	44.2
IMRNet	✓	**V**	PMSP+CTP	K400	3D ResNet-18	**76.8**	**45.0**

Table 2.4: **Results from the self-supervised setting**. Column `Compressed` indicates the methods that learn directly from compressed videos without decoding them. `Modality` indicates whether a method used only visual (**V**) modality or audio-visual modalities (**A+V**). `Pretrain` indicates datasets used for self-supervised pretraining. ‡: based on an official implementation by the authors. **Datasets**. K: Kinetics, AS: AudioSet, UCF: UCF101, HMDB: HMDB51.

version of our CTP task with only two modes: Aligned and Random (see Figure 2.4). Note that this is a typical pair correspondence setup used in the literature [3]. Table 2.4 (fourth group) shows the results. We can see that using either of our pretext tasks leads to a significant improvements compared to the `Scratch` result. The `CTP (Binary)` results suggests that the two additional transformation types (Shuffle and Shift in Figure 2.4) improves the task by making it more difficult to solve; we noticed that the loss curve of `CTP (Binary)` decreases significantly faster than `CTP` and quickly saturates thereafter.

2.3.3 Qualitative Examples

PMSP Label Visualization. In Section 2.2.2, we motivated the design of our PMSP task by arguing that there is an implicit *videographer bias* captured in videos in-the-wild that naturally reflects visual saliency: Videos are purposely recorded to highlight important objects and their movements; therefore, regions with the highest energy of motion – captured by our PMSP labels – can provide clues to learning video representation that captures salient moving objects. We acknowledge that this is, of course, a weak hypothesis. However, in this section we provide some convincing empirical evidence.

Figures 2.5-2.12 are generated by visualizing regions with the highest energy of motion – *i.e.* , the PMSP labels – at multiple spatio-temporal scales. It needs a bit of explanation on how to read the figures as there is a lot going on. Each figure is organized into three rows; each row shows results with multiple spatial regions at a particular temporal scale, $t \in \{1, 3, 5\}$. We color-code different spatial scales: Red boxes are in a $[2 \times 2]$ spatial scale, green boxes are in a $[3 \times 3]$ spatial scale, and blue boxes are in a $[4 \times 4]$ spatial scale. Notice that all five I-frames in the top rows ($t = 1$) in each set of results always contain identical regions. This is because, at the temporal scale $t = 1$ (meaning, a temporal grid of size 1), we compute the regions with the highest motion energy over the entire video (5 GOPs), hence the regions are identical across all I-frames in a video. Conversely, the bottom rows ($t = 5$, a temporal grid of size 5) show the regions computed at each GOP, and hence the regions may differ by every I-frame (recall that each GOP contains a single I-frame). The middle rows ($t = 3$) show regions computed over 3 GOPs. We overlay the regions at the overlapping I-frames, *e.g.* , the third I-frame at $t = 3$ contains regions computed at all three grid locations spanning over the I-frame indices [1,2,3], [2,3,4], and

[3,4,5]. We order the figures at an increasing level of complexity, and provide detailed analyses of the results in the captions of the figures.

The results in Figures 2.5-2.12 suggest that the most vibrant regions, as computed by our PMSP labels, tend to overlap with *semantically* important regions, *e.g.* , the most salient moving objects. Intuitively, training our model to detect those regions encourages it to learn visual representations that capture salient objects and motion. This allows our model to learn discriminative visual representation in a self-supervised manner.

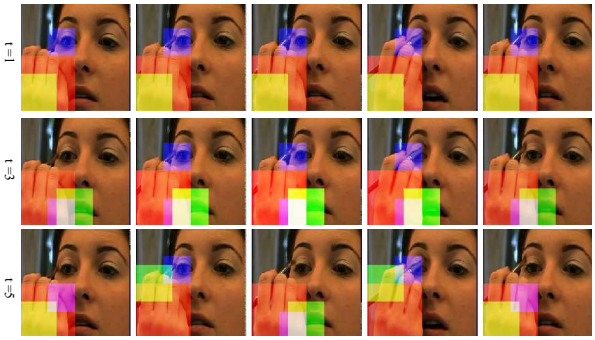


Figure 2.5: **PMSP label visualization**. The most vibrant regions, as highlighted by boxes of varying sizes indicating different spatial scales ([2×2], [3×3], [4×4]), all successfully capture the most salient moving object (the hand with a eye brush) and its motion (applying eye makeup).

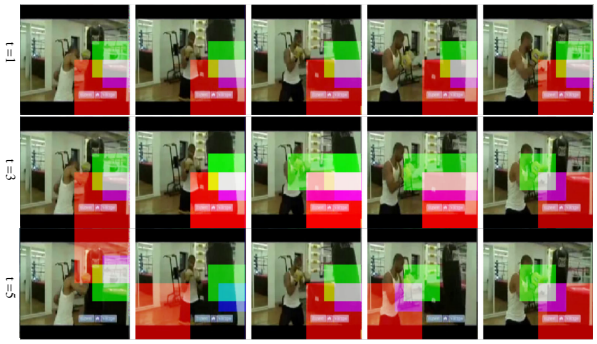


Figure 2.6: **PMSP label visualization**. At $t = 1$, the most vibrant region is the punching bag, correctly capturing the semantic category of the video (*BoxingPunchingBag*). As we get temporally finer, the regions start to capture the boxer's movement, *e.g.* , the elongated green boxes in the third and the fourth I-frames at $t = 3$.

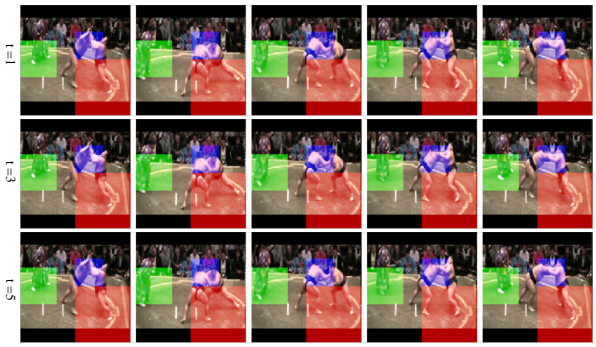


Figure 2.7: **PMSP label visualization**. This example highlights the benefit of formulating our task in a *spatially* pyramidal manner. Notice the boxes at different spatial scales capture different moving objects at varying sizes, i.e., green boxes ($[3 \times 3]$) capture the referee, blue boxes ($[4 \times 4]$) capture the hands of the two sumo wrestlers, and red boxes ($[2 \times 2]$) capture the wrestlers' legs.

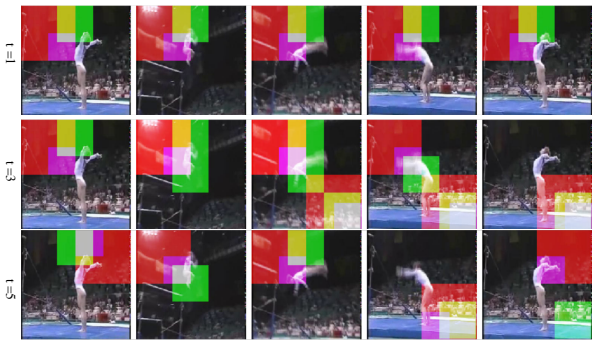


Figure 2.8: **PMSP label visualization**. This example highlights the benefit of formulating our task in a *temporally* pyramidal manner. While the vibrant regions at $t = 1$ fail to capture the tumbling gymnast, the vibrant regions at $t = 3$ and $t = 5$ successfully track her trajectory (especially the three middle I-frames at $t = 3$). In general, different videos will contain moving objects at different speeds; our pyramidal formulation allows us to capture a wide variety of moving objects at different speeds via multiple temporal scales.

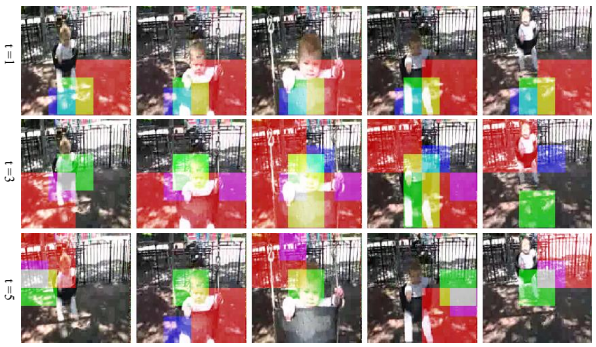


Figure 2.9: **PMSP label visualization**. Another example highlighting the benefit of our pyramidal formulation. While some regions at $t = 1$ miss the toddler in swing (see the first and the fifth I-frames in the first row), at $t = 3$ and $t = 5$ the boxes successfully track the toddler's trajectory.

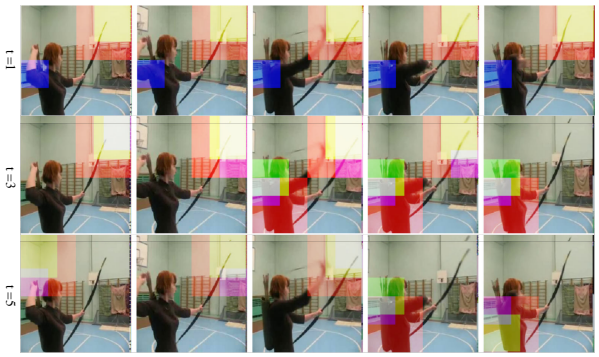


Figure 2.10: **PMSP label visualization**. This example contains a dynamically moving object (a women with a bow and an arrow) spanning across a large region in the frames, representing a challenging situation. Notice the boxes at different spatial and temporal scales highlight different parts: at $t = 1$, both the green ($[3 \times 3]$) and the red ($[2 \times 2]$) boxes capture the bow, which exhibits the sharpest edge with motion (hence the highest motion energy in those spatial scales), while the blue boxes ($[4 \times 4]$) capture the arm that takes out an arrow. At $t = 3$ and $t = 5$, the regions start to capture different parts, *e.g.* , the woman, which exhibits dynamic motion only towards the end of the video.

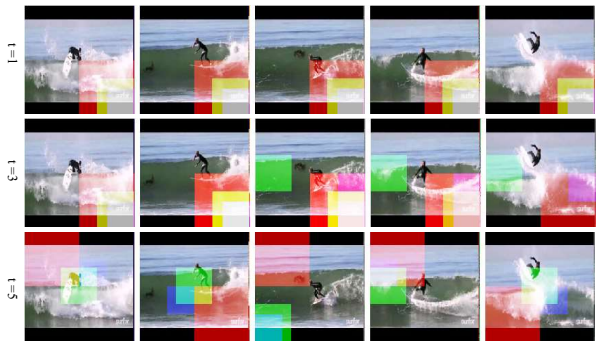


Figure 2.11: **PMSP label visualization**. Another challenging example containing a small, dynamically moving object (Surfing). At $t = 1$, all the boxes focus on the crushing waves on the bottom right corner, which is on average the most vibrant region in this video. Things are not much better at $t = 3$; the surfer is still too fast to capture, and thus the boxes fail to capture the surfer and instead highlight crushing waves. At the finest temporal scale $t = 5$, the boxes begin to capture the surfer (see the first, second and the fifth frames on the bottom).

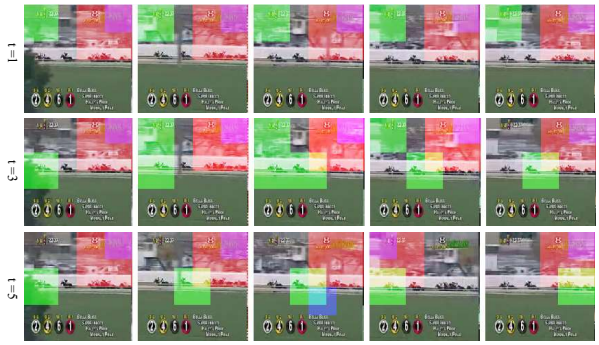


Figure 2.12: **PMSP label visualization**. This is a *partial*-failure example that shows boxes highlighting game statistics during horse race TV broadcast (see the boxes at $t = 1$). The game statistics constantly change frame-by-frame (*e.g.* , time marks, rank, etc.), which caused those regions to exhibit on average the highest energy of motion for the entire duration of the video. Learning representations that strictly focus on those regions could lead to non-discriminative information (many sports videos show similar game statistics on screen). Fortunately, the boxes begin to highlight the horse riders at a finer temporal scale; see the green boxes ($[3 \times 3]$) at $t = 3$ and $t = 5$. This, again, suggests that the pyramidal formulation makes our PMSP task robust to a variety of challenging real-world scenarios.

PMSP Prediction Results. Figure 2.13 and Figure 2.14 show side-by-side comparisons of the ground-truth PMSP labels and our prediction results. Overall, our prediction results are mostly identical to the ground-truth regions. When our prediction deviates from the ground-truth, the predicted regions still tend to capture important moving objects, e.g., Figure 2.13 (a) captures different parts of the punching bag, Figure 2.14 (b) captures different parts of the boxer, and Figure 2.14 (d) captures different arms of the swimmer.

Video-to-Video Retrieval. To demonstrate the quality of video representations learned using our self-supervised learning objectives (Section 2.2.2), we evaluate our method in the video-to-video retrieval task. To do this, we measure the cosine similarity between a query video and all the other videos in a candidate set, and show the top-1 retrieved video. We compare ours to two baselines: `3D Rotation` [37] is our IMRNet pretrained using the 3D rotation prediction task (we used the *IMRNet + Rotation* pretrained model reported in Table 2.4), and `ImageNet` is a ResNet-152 fully-supervised with ImageNet ILSVRC-2012 [88]. We visualize the results in Figures 2.15-2.18 and analyze the results in the caption of each figure.

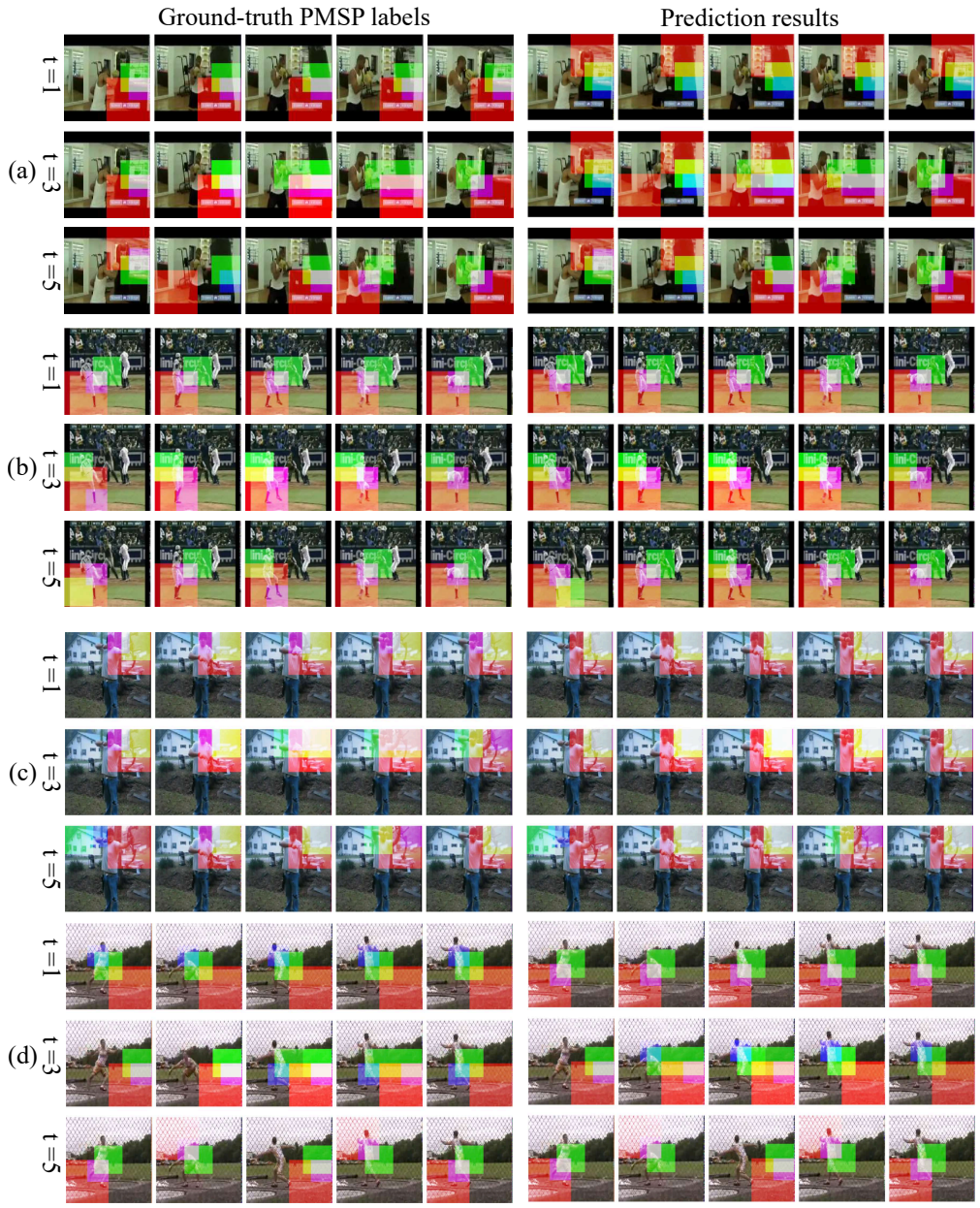


Figure 2.13: **PMSP prediction results**. Overall, the predicted regions tend to highlight salient moving objects (although sometimes different from the ground-truth). (a): BoxingPunchingBag, (b): BaseballPitch, (c): Archery, (d): ThrowDiscus.

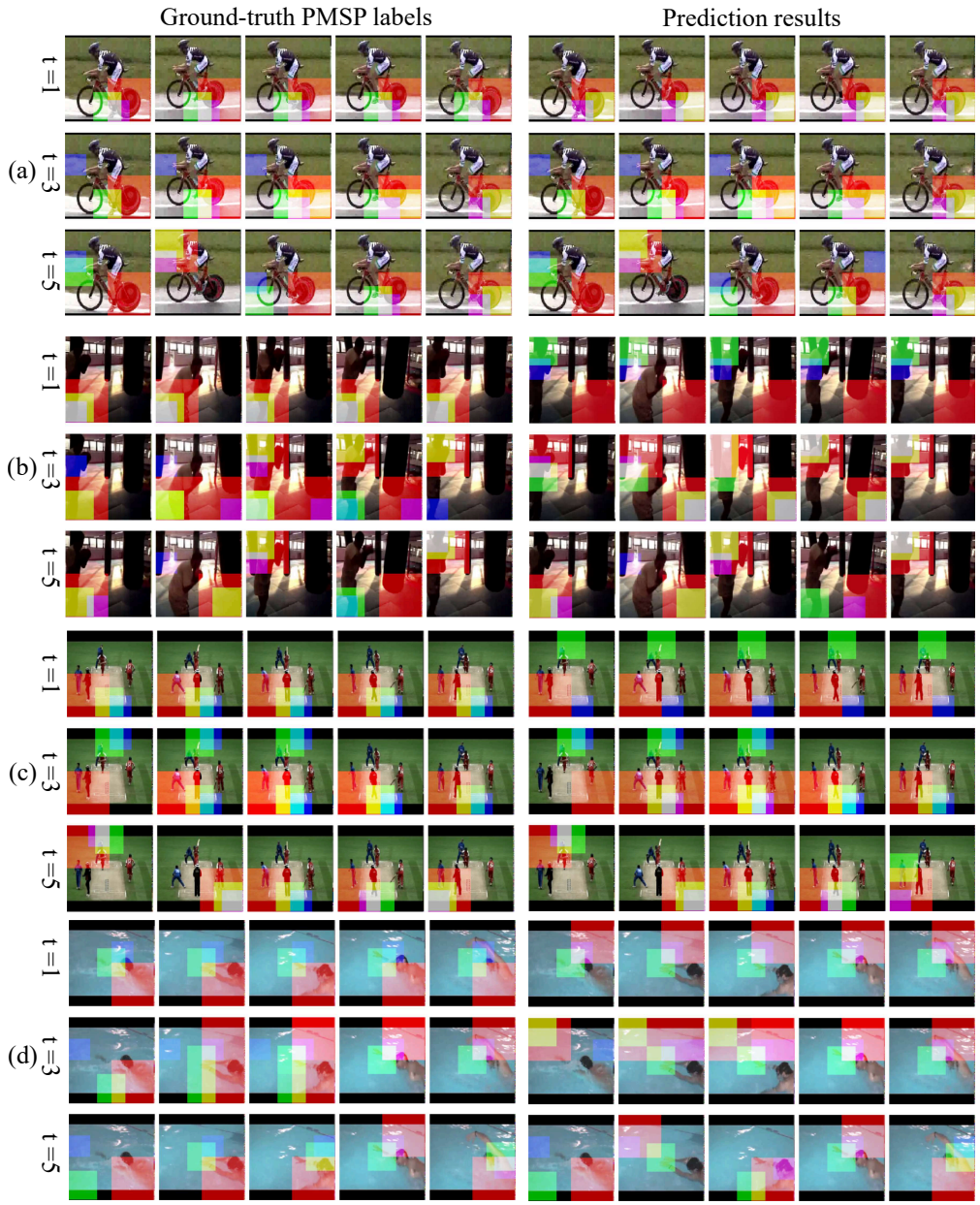


Figure 2.14: **PMSP prediction results**. Overall, the predicted regions contain the salient moving objects (although sometimes different from the ground-truth). (a): Biking, (b): BoxingPunchingBag, (c): CricketShot, (d): FrontCrawl.

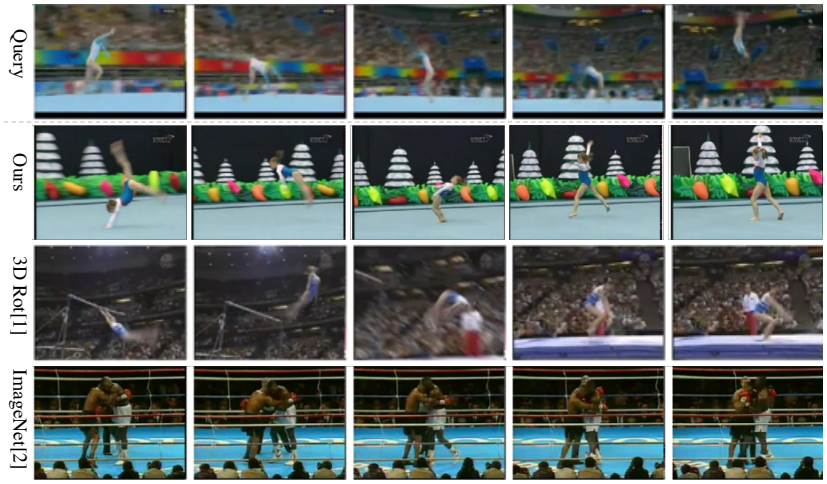


Figure 2.15: **Video-to-video retrieval result**. Ours finds the most similar video to the query in terms of both the appearance (a gymnast) and the motion (handspring). The 3D Rotation baseline captures perhaps more similar appearance (a gymnast with the audience in the back) but less similar motion (horizontal bar jump vs. handspring). The ImageNet baseline fails to capture both appearance and motion (ImageNet does not contain a category relevant to floor gymnastics).

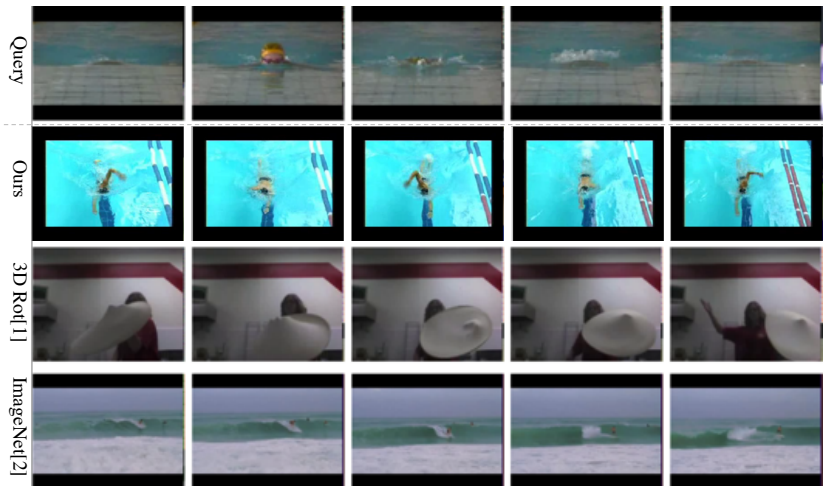


Figure 2.16: **Video-to-video retrieval result**. Ours finds the most similar video to the query in terms of both the appearance (swim stadium) and motion (swimming). The ImageNet baseline does capture similar appearance (water), but fails to capture motion (swimming vs. surfing). The 3D Rotation baseline shows little to no semantic similarity to the query video.

Figure 2.17: **Video-to-video retrieval result**. Ours finds the most similar video to the query in terms of both the appearance (scene layout) and the motion (pitching). The ImageNet baseline does capture similar high-level semantics appearance-wise (baseball pitcher) but motion is relatively less similar (different camera angle, no catcher and no hitter). The 3D Rotation baseline shows little to no semantic similarity to the query video.

Figure 2.18: **Video-to-video retrieval result.** All three retrieval results fail to find videos that belong to the same semantic category as the query video (pole vault). However, ours finds a video that contains similar appearance (running track) and similar motion (running and jumping). The ImageNet baseline also captures similar appearance (javelin throw) but less similar motion (running at a substantially slower pace). The 3D Rotation baseline shows little to no semantic similarity to the query video.

2.4 Related Work

Self-Supervised Learning of Video Representation. Self-supervised learning has received significant attention [89, 90, 91, 30]. Based on strong progress in the image domain, several works proposed to learn video representations in a self-supervised manner. One popular idea is leveraging temporal information [30, 31, 32, 25, 33, 34]. Temporal coherence of video pixels has been leveraged as a self-supervisory signal [92, 93]. Another popular idea is learning transformation-invariant representations [35, 36, 37]. Also, contrastive learning [94, 95, 82, 81] has been successfully applied to videos [87]. Despite active research in this field, to the best of our knowledge, there has not been prior work on self-supervised learning from compressed videos.

Compressed Video Recognition. Compressed video understanding has been tackled in a supervised setting [50, 51, 52]. Existing approaches encode each stream separately and perform late fusion, *e.g.* , feature concatenation [50, 51]. However, as we show in our experiments, this can miss out useful information that can only be learned by modeling the interaction across streams. Unlike previous approaches, our approach shares relevant information across streams during the encoding process. In addition, because compressed videos do not provide continuous RGB frames, it is not easy to directly apply 3D CNNs to encode I-frames. Therefore, existing approaches use 2D CNNs to process compressed video frames, e.g., CoViAR [51] uses 2D CNNs to process each stream and perform average pooling over P-frames, which is insufficient to model complex motion dynamics. DMC-Net [52] reconstructs the optical flow from P-frames and later use the reconstructed signal as input to I3D [76], but this requires ground-truth optical flow which is compute-intensive. Instead, our IMR network adopts the gated attention [96, 97] and bidirectional connection [98, 11]

for lateral connection to model complex motion dynamics with I,P frames freely available in compressed videos.

2.5 Conclusion

We introduced an IMR network for compressed video recognition and two pretext tasks for self-supervised learning of compressed video representation. Our work complements and extends existing work on compressed video recognition by (1) proposing the first self-supervised training approach on the compressed videos, and (2) proposing a three-stream 3D CNN architecture to encode compressed videos while dynamically modeling interaction between I-frames and P-frames. We demonstrated that our IMRNet outperforms state-of-the-art approaches for compressed videos in both fully-supervised and self-supervised settings, and that our pretext tasks yield better performance in downstream tasks.

Chapter 3

Parameter Sharing Schemes for Multimodal Transformers

3.1 Introduction

Learning multimodal representation from unlabeled videos has received considerable attention [99]. Audio-visual learning is of particular interest due to the abundance of videos with natural audio-visual co-occurrence [39, 100, 101, 102, 103, 104]. However, existing approaches learn localized representations from short videos (hundreds of milliseconds to just under a few seconds), capturing only *short-term* dependencies in data. While this is useful for certain applications, e.g., source separation [102] and atomic action recognition [105], learning representation that captures *long-term* dependencies is equally important, e.g., for activity recognition [71, 106, 107]. Unfortunately, processing long videos requires large memory resource and capturing long-term dependencies is a long-standing problem [108, 109, 54].

In language understanding, strong progress has been made in large-scale

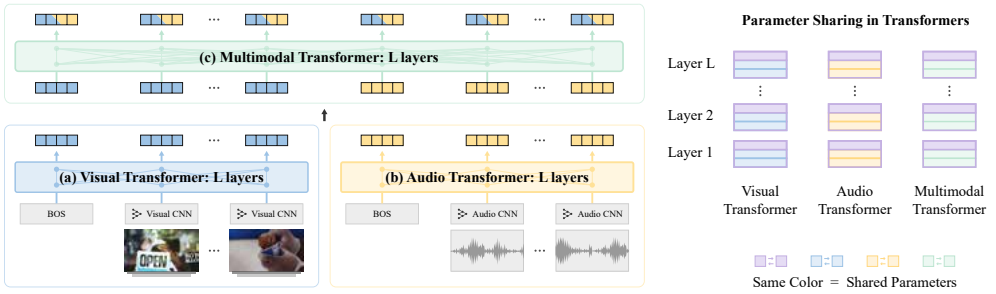


Figure 3.1: **(Left)** Our model consists of CNNs encoding short-term dynamics of each modality and Transformers encoding long-term dynamics of audio-visual information from videos. **(Right)** To alleviate excessive memory requirements, we propose an efficient parameter sharing scheme based on matrix decomposition with low-rank approximation, which allows us to train our model end-to-end.

learning of contextualized language representations using Transformers [54, 55, 56, 57, 58, 46, 59, 60]. Riding on the success of Transformers, several recent works have extended it to the multimodal setting by adding an additional vision module to the Transformer framework [61, 110]. However, these models are typically not end-to-end trained; they rely on a language-pretrained BERT [46], which is fixed throughout, and train only the visual components. While the pretrained BERT helps accelerate convergence and brings reliable extra supervision signal to the vision component, this partial learning setup can be undesirable if the text data comes from different distributions (of topics, dialects, or foreign languages) or if we want to apply it to different modalities (e.g., audio-visual). Unfortunately, end-to-end training of such multimodal Transformer architectures is challenging for most existing compute environments due to the excessive memory requirement.

In this work, we make three key contributions. First, we propose an *end-to-end trainable* bidirectional transformer architecture that learns contextualized audio-visual representations of long videos. Our model, shown in Figure 3.1,

consists of audio/visual CNNs, audio/visual Transformers, and a multimodal Transformer. The CNNs operate on short (e.g., one second) video clips and are intended to capture short-term dynamics within each modality. The Transformer layers operate on long video sequences (e.g., 30 seconds), capturing long-term dynamics. To enable end-to-end training, we propose a novel parameter reduction technique that shares parts of weight parameters across Transformers and across layers within each Transformer. We show that this results in up to 97% parameter reduction, enabling end-to-end training of our model, with a minimal performance degradation. To the best of our knowledge, our work is the first to report end-to-end trained multimodal Transformers, and the first to apply Transformers for audio-visual representation learning.

The quality of negative samples is crucial in contrastive learning, which is part of our learning objective. As our second contribution, we propose a *content-aware* negative sampling strategy that favors negatives sufficiently similar to a positive instance. Our approach measures the similarity by reusing the CNN embeddings obtained during model training, and thus do not introduce extra parameters to learn. We show that this improves performance over the standard sampling strategies.

Our third contribution is a systematic evaluation of different modality fusion strategies. Existing works on multimodal BERT (all using vision-and-language data) typically apply one fusion strategy without thoroughly comparing with alternatives, e.g., some works perform early fusion [61, 111] while others perform mid-level fusion [110, 112]. As a result, it is unclear how different fusion methods affect the final performance. In this work, we compare three fusion strategies (early, mid, late) and show the superiority of mid-level fusion.

To demonstrate our approach, we pretrain our model on long (30-second) video clips from Kinetics-700 [106] and finetune it on various video classification

tasks. One benefit of the modular design of our architecture is flexibility: once pretrained, we can use any of the subnetworks for downstream tasks depending on the modalities involved (audio-only, visual-only, audio-visual) and video lengths (short and long). To show this, we evaluate our model on UCF101 [1] and ESC-50 [5] for short-term visual/audio classification, and Charades [107] and Kinetics-Sounds [3] for long-term audio-visual action recognition.

3.2 Approach

Figure 3.1 shows an overview of the proposed model architecture. The input to our model is a sequence of visual clips $\mathbf{v}_{1:T}$ and the corresponding sequence of audio streams $\mathbf{a}_{1:T}$. For example, each sequence is a 30 second-long video divided into 30 non-overlapping clips (each clip is one second long). We divide our model into three parts with different characteristics, which are explained below.

Local Feature Embedding. We feed each of T video clips to a visual CNN $f_V(\mathbf{v}_t)$ to obtain $\mathbf{x}^v_{1:T} \in \mathbb{R}^{T\times D}$, and each audio stream to an audio CNN $f_A(\mathbf{a}_t)$ to obtain $\mathbf{x}^a_{1:T} \in \mathbb{R}^{T\times D}$. [1] Intuitively, the CNN outputs are *temporally local* embeddings as they have access to only a short-range temporal window of the entire video sequence. Thus, they are suitable for representing short-range atomic actions (e.g., sit down, raise arms) that constitute long-range events (e.g., gym workout). We use the SlowFast network [11] with a ResNet-50 backbone [74] as a visual CNN f_V, and a ResNet-50 as an audio CNN f_A. The weights of both CNNs are randomly initialized and trained end-to-end with the Transformer layers.

Unimodal Contextualized Embedding. The local feature embeddings

[1] For notational simplicity, we drop the subscripts to refer to the entire sequence unless distinction is necessary.

capture short-term dynamics but lack long-term contextual information. We use Transformers [54] to enrich the embeddings with sequence-level context. We start by learning *unimodal* contextualized representations using the visual Transformer g_V and the audio Transformer g_A, respectively.

The Transformer consists of L layers, each with two sub-layers: a multi-head attention layer and a feed-forward layer. Given an input sequence of embeddings $\mathbf{x} \in \mathbb{R}^{T \times D}$ and A attention heads, the j-th head in the attention layer computes the output embedding sequence $\mathbf{a}_j \in \mathbb{R}^{T \times \gamma}, \gamma = D/A$ as

$$\mathbf{a}_j = \text{softmax}\left(\frac{Q_j K_j^\top}{\sqrt{\gamma}}\right) V_j, \qquad Q_j = \mathbf{x}W_j^q, K_j = \mathbf{x}W_j^k, V_j = \mathbf{x}W_j^v \tag{3.1}$$

where $W_j^q, W_j^k, W_j^v \in \mathbb{R}^{D \times \gamma}$ are weight matrices for computing the (`query`, `key`, `value`) triplet given the input $\mathbf{x}$. This operation is repeated for each attention head, and the outputs are combined (with concatenation followed by one linear layer with weights $W^b \in \mathbb{R}^{D \times D}$), producing $\mathbf{a} \in \mathbb{R}^{T \times D}$. Next, the feed-forward layer takes this intermediate output and computes $\mathbf{o} \in \mathbb{R}^{T \times D}$ using a two-layer fully-connected network with weights $W^c \in \mathbb{R}^{D \times E}$ and $W^d \in \mathbb{R}^{E \times D}$. The output of each sub-layer is computed using a residual function followed by layer normalization [113], i.e., $\text{LayerNorm}(x + \text{Sublayer}(x))$. In this work, we set the number of layers $L = 6$, the number of attention heads $A = 12$, the feature dimension $D = 768$ and the intermediate dimension $E = 3072$. For simplicity, we use this design for all layers across all three Transformers in our model.

Before feeding local embeddings $\mathbf{x}^v$ and $\mathbf{x}^a$ to unimodal Transformers, we augment them with "positional" embeddings. Specifically, we append to the beginning of each sequence a special vector `BOS` (beginning of sequence), i.e., $\mathbf{x}_0^v$ for visual and $\mathbf{x}_0^a$ for audio streams; their dimensions are same as $\mathbf{x}_t^v$ and $\mathbf{x}_t^a$, respectively. We also define positional embeddings $\mathbf{p}_{0:T}$ encoding time indices (we call this "time" embedding). This is necessary to preserve information

about temporal ordering of local feature embeddings, which is otherwise lost in Equation 3.1. We combine them via layer normalization,

$$\mathbf{u}_t^v = \text{LayerNorm}(\mathbf{x}_t^v + \mathbf{p}_t^v), \quad \mathbf{u}_t^a = \text{LayerNorm}(\mathbf{x}_t^a + \mathbf{p}_t^a), \quad \forall t \in [0, T] \quad (3.2)$$

We initialize $\{\mathbf{x}_0^v, \mathbf{x}_0^a, \mathbf{p}_{0:T}^v, \mathbf{p}_{0:T}^a\}$ to the normal distribution and train them with the rest of the model.

We feed the augmented visual embeddings into the visual Transformer g_V and obtain $\mathbf{y}_{0:T}^v = g_V(\mathbf{u}_{0:T}^v)$, and similarly obtain $\mathbf{y}_{0:T}^a = g_A(\mathbf{u}_{0:T}^a)$. The embeddings at each time step has a direct access to the entire input sequence regardless of their position (it has a one-step signal path during forward and backward inference). Multiple layers of such feature transformation thus allow the resulting embedding to be deeply contextualized in the time dimension. We denote the output embeddings corresponding to the `BOS` positions by $\texttt{BOS}_g^v = \mathbf{y}_0^v$ and $\texttt{BOS}_g^a = \mathbf{y}_0^a$, and designate them as the *summary* embeddings representing the sequence of each modality.

Multimodal Contextualized Embedding. The unimodal embeddings capture long-term temporal context but miss out on cross-modal information. The final step in forward inference is to use a multimodal Transformer h_{AV} to obtain embeddings contextualized in the audio-visual space.

We first augment the embeddings $\mathbf{y}_{0:T}^v$ and $\mathbf{y}_{0:T}^a$ with modality and time embeddings. The modality embeddings $\mathbf{m}^v$ and $\mathbf{m}^a$ are vectors of the same dimension as $\mathbf{y}_t^v$ and $\mathbf{y}_t^a$, respectively. We share $\mathbf{m}^v$ (and $\mathbf{m}^a$) across all the unimodal embeddings $\mathbf{y}_{0:T}^v$ (and $\mathbf{y}_{0:T}^a$); thus, they add modality-discriminative information to the Transformer. We also add time embeddings $\mathbf{p}_{0:T}$ as before; however, unlike in the previous step, we share the same $\mathbf{p}_{0:T}$ between embeddings from the two modalities to correctly indicate the time indices. We augment

the modality and time embeddings via layer normalization,

$$\mathbf{w}_t^v = \text{LayerNorm}(\mathbf{y}_t^v + \mathbf{p}_t + \mathbf{m}^v),\ \mathbf{w}_t^a = \text{LayerNorm}(\mathbf{y}_t^a + \mathbf{p}_t + \mathbf{m}^a),\ \forall t \in [0, T] \tag{3.3}$$

We feed the augmented visual embeddings $\mathbf{w}_{0:T}^v$ and audio embeddings $\mathbf{w}_{0:T}^a$ to the multimodal Transformer h_{AV}, one after another, and obtain $\mathbf{z}_{0:(2T+1)} = h_{AV}([\mathbf{w}_{0:T}^v; \mathbf{w}_{0:T}^a])$. We again denote the output embeddings corresponding to the `BOS` positions by $\texttt{BOS}_h^v = \mathbf{z}_0^v (= \mathbf{z}_0)$ and $\texttt{BOS}_h^a = \mathbf{z}_0^a (= \mathbf{z}_{T+1})$, and use them as summary embeddings encoding multimodal context.

We emphasize the importance of feeding $\mathbf{w}_{0:T}^v$ and $\mathbf{w}_{0:T}^a$ one after another. An alternative would be concatenating them before feeding them to h_{AV} and obtaining an output $\mathbf{z}_{0:T}$ (instead of $\mathbf{z}_{0:(2T+1)}$). However, this restricts the Transformer to access audio-visual embeddings only from the same time slices, which could be problematic when there is a temporally asynchronous relationship between the two modalities (e.g., a visual clip matches with sound captured a few times steps before) [114, 42]. By arranging the two sequences one after the other, the Transformer can mix-and-match appropriate audio-visual embeddings in an asynchronous manner. Another practical concern with the alternative approach is that it significantly increases the model size; the weight matrices W_q, W_k, W_v grow quadratically with the input feature dimension D. Serializing the input resolves both issues.

3.2.1 Self-Supervised Pretraining Objectives

Task 1: Masked Embedding Prediction (MEP). BERT [46] is trained using the masked language model (MLM) task, which randomly selects input tokens and replaces them with a mask token. The model is then trained to predict the original (unmasked) tokens by solving a classification task with a cross-entropy loss. However, inputs to our model are real-valued audio-visual

signals (rather than discrete tokens),[2] so applying the MLM task requires input discretization, which causes information loss [110, 115]. We instead train our model to identify the correct visual clip or audio stream compared to a set of negative samples in a contrastive manner, which does not require input discretization.

We formulate our MEP task using InfoNCE [94], which is the softmax version of the noise contrastive estimation (NCE) [116]. Let $\tilde{\mathbf{o}}_t$ be the t-th output of any of the three Transformers obtained by masking the t-th input $\mathbf{x}_t$. Our InfoNCE loss is then defined as

$$\mathcal{L}_{\text{NCE}}(\mathbf{x}, \tilde{\mathbf{o}}) = -\mathbb{E}_{\mathbf{x}} \left[\sum_t \log \frac{\text{I}(\mathbf{x}_t, \tilde{\mathbf{o}}_t)}{\text{I}(\mathbf{x}_t, \tilde{\mathbf{o}}_t) + \sum_{j \in \text{neg}(t)} \text{I}(\mathbf{x}_j, \tilde{\mathbf{o}}_t)} \right], \tag{3.4}$$

where $\text{neg}(t)$ are negative sample indices and the compatibility function $\text{I}(\mathbf{x}_t, \tilde{\mathbf{o}}_t)$ is,

$$\text{I}(\mathbf{x}_t, \tilde{\mathbf{o}}_t) = \exp\left(\text{FFN}^\top(\tilde{\mathbf{o}}_t) W_I \mathbf{x}_t \right), \tag{3.5}$$

where $W_I \in \mathbb{R}^{P \times D}$ ($P = 256$) and FFN is a two-layer feed-forward network. The use of a non-linear prediction head has shown to improve the quality of the representations learned in a contrastive learning setup [81]; following the recent work in Transformers [46, 59, 117], we use a GELU non-linear activation function [118] in FFN. Optimizing Equation 3.4 enforces $\text{I}(\mathbf{x}_t, \tilde{\mathbf{o}}_t)$ to approximate the density ratio $\frac{p(\mathbf{x}_t|\tilde{\mathbf{o}}_t)}{p(\mathbf{x}_t)}$; this can be seen as maximizing the mutual information between $\mathbf{x}_t$ and $\tilde{\mathbf{o}}_t$ [94]. Intuitively, this encourages the Transformer to capture the underlying dynamics of $\mathbf{x}$ from each modality without explicitly learning a generative model $p(\mathbf{x}_t|\tilde{\mathbf{o}}_t)$.

Negative Sampling. We find that a good negative sampling strategy is essential for the model's convergence. Existing approaches either use all but $\mathbf{x}_t$

[2] In the form of RGB images and log-mel-scaled spectrograms.

(positive) within a mini-batch as negative samples or limit it to the current sequence only. However, both these methods ignore the data content and thus can miss useful negatives. Oord et al. [94] showed that leveraging prior knowledge about data can improve the negative sample quality (e.g., by sampling negatives from the same speaker as the positive). Unfortunately, such prior knowledge is often not available in unlabeled videos.

We propose a *content-aware* negative sampling strategy that favors negatives sufficiently *similar* to a positive instance in the CNN embedding space; we call our approach `CANS-Similar`. Our approach is inspired by Ulyanov et al. [119] who showed that randomly initialized CNNs provide a strong prior over natural images due to the inductive bias already built into the design of the CNNs. This suggests that our local feature embeddings $\mathbf{x}^v$ (and $\mathbf{x}^a$) can capture the underlying statistical regularities in video clips (and audio streams) right from the beginning, which can be sufficient to assess the similarity/dissimilarity between clips. Therefore, the distance measured on them can approximate content dissimilarity well (and this will improve as the training progresses).

Motivated by this, we sample the negatives based on local feature embeddings $\mathbf{x}^v$ (and $\mathbf{x}^a$). Specifically, we compute a pairwise ℓ_2 distance between $\mathbf{x}_t$ (positive) and all other instances within a mini-batch, and normalize them to the [0, 1] interval. To remove samples that are either too similar or too different from the positive sample, we discard instances that fall outside the 95% confidence interval in the normalized distance space. We then sample the negatives from the remainder using the normalized distance as sampling probability. This makes instances similar to the positive instance have more chance to become negatives. We emphasize the importance of sampling, instead of deterministically taking top most similar samples; the stochasticity allows our model to be robust to potentially inaccurate distance estimates because samples with low

probabilities will still have a chance to be selected as negatives.

Finally, our MEP loss is the InfoNCE loss computed on all three Transformers,

$$\mathcal{L}_{\text{MEP}} = \mathcal{L}_{\text{NCE}}(\mathbf{x}^a, \tilde{\mathbf{y}}^a) + \mathcal{L}_{\text{NCE}}(\mathbf{x}^v, \tilde{\mathbf{y}}^v) + \mathcal{L}_{\text{NCE}}([\mathbf{x}^a; \mathbf{x}^v], \tilde{\mathbf{z}}) \tag{3.6}$$

Task 2: Correct Pair Prediction (CPP). The MEP task encourages our model to learn the underlying dynamics within each modality. To help our model learn cross-modal dynamics, we design a task that predicts whether a pair of audio-visual embeddings is from the same video. Specifically, we define two binary classifiers, one for the two unimodal Transformers and another for the multimodal Transformer. Each classifier takes as input either $\mathbf{s}_g = [\mathbf{y}_0^v; \mathbf{y}_0^a]$ (and $[\mathbf{z}_0^v; \mathbf{z}_0^a]$), a pair of audio-visual "summary" embeddings corresponding to the `BOS` positions, or $\mathbf{s}_h = [\mathbf{y}_t^v; \mathbf{y}_t^a]$ (or $[\mathbf{z}_t^v; \mathbf{z}_t^a]$), the output embeddings sampled at random positions (we take two random positions $t \in [1, T]$). The classifier predicts $p(c|\mathbf{s})$ indicating whether the pair is from the same video ($c = 1$) or from different videos ($c = 0$). We train the classifiers with a binary cross-entropy loss,

$$\mathcal{L}_{CPP} = -\mathbb{E}_{\mathbf{x},\mathbf{y}} \left[c \cdot \log p(c|\mathbf{s}_g) + c \cdot \log p(c|\mathbf{s}_h) \right] \tag{3.7}$$

where $\cdot$ is the inner product. We generate a random derangement of the input mini-batch so that the number of positive and negative pairs are guaranteed to be the same.

Overall Pretraining Objective. We train our model end-to-end from scratch by optimizing $\mathcal{L}_{MEP} + \alpha \mathcal{L}_{CPP}$ with a balancing term α. We find our model is insensitive to this term, so we set $\alpha = 1.0$.

3.2.2 Parameter Reduction

Optimizing our model is challenging due to the large memory requirement. The most expensive part is the Transformers, which take up 82% of model param-

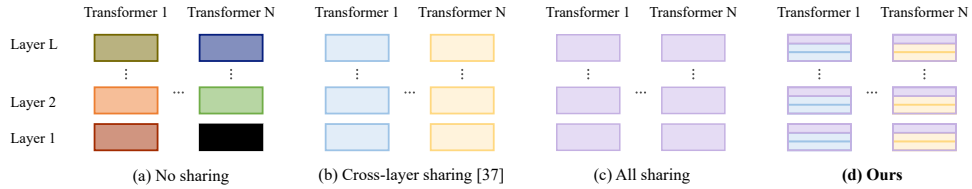


Figure 3.2: Comparison of parameter sharing schemes. Ours combines (b) and (c) but decomposes weights in each layer into private and shared parts so only the latter is shared across Transformers.

eters. One could reduce the model size by making the Transformers shallower, but the depth of Transformers has shown to be crucial to get good performance [46]. We propose to reduce the model size by aggressively sharing parts of weights across Transformers as well as layers within each Transformer (see Figure 3.2 (d)).

Sharing across Transformers. We first consider sharing weights across Transformers. Each Transformer encodes data coming from different distributions: g_V encodes $\mathbf{x}^v$, g_A encodes $\mathbf{x}^a$, and h_{AV} encodes $(\mathbf{y}^v, \mathbf{y}^a)$. These input distributions may each exhibit different dynamics, yet together share certain regularities because they all come from the same videos. Motivated by this, we decompose Transformer weights into shared and private parts so that different patterns can be learned in a parameter-efficient manner. Recall that each layer of a Transformer contains weights $\{W^q, W^k, W^v, W^b, W^c, W^d\}$. We decompose each of these weights into $W = U\Sigma V^\top$, where $W \in \mathbb{R}^{M\times N}, U \in \mathbb{R}^{M\times O}, \Sigma \in \mathbb{R}^{O\times O}, V \in \mathbb{R}^{N\times O}$. We perform low-rank approximation of W by setting the rank $O \ll M, N$, and share U across Transformers while keeping Σ and V private to each Transformer. This helps reduce parameters because $MO + 3(O^2 + NO) \ll 3MN$. We experimented with different matrix ranks O but the differences were small; we set $O = 128$ ($M, N = 768$ or 3072).

The decomposition converts a linear projection of input $W\mathbf{x}$ into a series of (unconstrained) linear projections $U\Sigma V^\top \mathbf{x}$. However, this can cause numerical instability during optimization [120]. We could perform the Singular Value Decomposition (SVD) over W so that it performs rotation ($V^\top$), stretch (Σ), and rotation (U) with orthogonal basis vectors in U and V. Unfortunately, solving the full SVD has a computational complexity of $\mathcal{O}(\max(M, N)^2)$ [121]. Here, we put an orthogonality constraint only on Σ and perform projection ($V^\top$), rotation (Σ), and projection (U) of input $\mathbf{x}$. In addition, we put $V^\top \mathbf{x}$ in a unit sphere (via ℓ_2-normalization) before rotating it with Σ. This not only improves numerical stability, but also removes magnitude information in $V^\top \mathbf{x}$ and keeps angular information only, which has been shown to provide sample discriminative information [122]. To impose the orthogonality constraint on Σ, we use the Padé approximation with a scale-squaring trick of Lezcano-Casado and Martínez-Rubio [123]. Intuitively, we linearly project $\mathbf{x}$ onto a unit sphere ($V^\top \mathbf{x}$) and rotate it ($\Sigma V^\top \mathbf{x}$) in each Transformer so that it captures the dynamics of each input distribution independently. We then project it to the shared space via U, capturing shared regularities across all three Transformers.

Sharing across Layers. Recently, Bai et al. [124] showed that sharing parameters across layers in deep neural networks does not hurt the representational power of the network. Furthermore, Lan et al. [117] demonstrated that cross-layer parameter sharing in the Transformer leads to a lighter and faster-to-train model without sacrificing the performance on various language understanding benchmarks. Motivated by this, we let each Transformer share parameters across different layers.

3.3 Experiments

We pretrain our model on Kinetics-700 [106] or AudioSet [5] and finetune it on various downstream tasks. The official release of Kinetics-700 contains 10-second clips only, so we download 410K original videos from YouTube and take 30-second clips from each video. For fair comparison with prior work, we use 10-second clips from the official release of AudioSet (we used 1.8M clips).

We pretrain our model on 64 NVIDIA Tesla V100 GPUs with a batch size of 256 for 220K iterations. For downstream tasks, we evaluate on *short*-video/audio classification using UCF101 [1] (13K clips from 101 classes; 7.2 seconds on average) and ESC-50 [5] (2K clips from 50 classes; 5 seconds), and on *long*-video classification using Kinetics-Sounds [3] (23K videos from 32 classes; 10 seconds on average) and Charades [107] (10K videos from 157 classes; 30 seconds on average).

3.3.1 Experimental Setup

Architectures of Visual/Audio CNNs. Table 3.1 shows the architectures of visual and audio CNNs we use for our model, that is, the SlowFast network [11] with a ResNet-50 backbone [74] and a ResNet-50, respectively. For the SlowFast network, we use the speed ratio $\alpha = 8$ and the channel ratio $\beta = 1/8$ for the SlowFast architecture, so $T_f = 8 \times T_s$. we use different values of T_s and T_f for different tasks. During pretraining, we set $T_s = 4$ and $T_f = 32$. During finetuning, we use $T_s = 8$ and $T_f = 64$ for short-video action classification on UCF101 while we use $T_s = 4$ and $T_f = 32$ for long-video action classification on Charades and Kinetics-Sounds. For the audio ResNet-50, we omit the down-sampling layer pool_1 to preserve information along both frequency and time axis in early stages. We use different values of T_a for different training phases. We set $T_a = 220$ for one-second clip during pretraining while we use $T_a = 440$

Stage	Visual CNN		Audio CNN
	Slow pathway	*Fast* pathway	
raw clip	$3 \times T_s \times 112^2$	$3 \times T_f \times 112^2$	$128 \times T_a$
conv_1	$1 \times 7^2, 64$ stride $1, 2^2$	$5 \times 7^2, 8$ stride $1, 2^2$	$9 \times 9, 32$ stride $1, 1$
pool_1	1×3^2, max stride $1, 2^2$	1×3^2, max stride $1, 2^2$	–
res_2	$\begin{bmatrix} 1 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 8 \\ 1 \times 3^2, 8 \\ 1 \times 1^2, 32 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 32 \\ 3 \times 3, 32 \\ 1 \times 1, 128 \end{bmatrix} \times 3$
res_3	$\begin{bmatrix} 1 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 1^2, 16 \\ 1 \times 3^2, 16 \\ 1 \times 1^2, 64 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 4$
res_4	$\begin{bmatrix} 3 \times 1^2, 256 \\ 1 \times 3^2, 256 \\ 1 \times 1^2, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 1^2, 32 \\ 1 \times 3^2, 32 \\ 1 \times 1^2, 128 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 6$
res_5	$\begin{bmatrix} 3 \times 1^2, 512 \\ 1 \times 3^2, 512 \\ 1 \times 1^2, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 3$

Table 3.1: The architectures of visual and audio CNNs. For the visual CNN, the input dimensions are denoted by {*channel size, temporal size, spatial size*^{2}}, kernels are denoted by {*temporal size, spatial size*^{2}*, channel size*} and strides are denoted by {*temporal stride, spatial stride*^{2}}. For the audio CNN, the input dimensions are denoted by {*frequency size, temporal size*}, kernels are denoted by {*frequency size, time size, channel size*} and strides are denoted by {*frequency stride, temporal stride*}.

for two-second clip during finetuning.

Data Preprocessing. We preprocess the data by dividing T-second clips into T non-overlapping parts ($T = 30$ for Kinetics-700 and $T = 10$ for AudioSet) and sampling 16 frames from each. For audio stream, we take waveform sampled at 44.1 kHz and convert it to log-mel-scaled spectrogram. We augment audio data with random frequency/time masking using SpecAugment [125], and visual data with color normalization, random resizing, random horizontal flip, and random cropping to obtain 112×112 pixel frames; for test data, we resize

videos to 128 pixels on the shorter side and take three equidistant crops of 128×128 pixels to cover the entire region. We also apply audio-visual synchronized temporal jittering [29].

Downstream Evaluation. For evaluation on UCF101, we follow the test protocol of Feichtenhofer et al. [11]: We sample 10 clips from each test video at a uniform time interval, and for each sampled clip, we take three equidistant spatial crops, resulting in a total of 30 views. We use each of the 30 views as input to our visual CNN and average the prediction scores from all 30 views to obtain the final prediction result. For evaluation on ESC-50, we extract 10 equally spaced 2-second clips from each test audio sample. We use each of 10 clips as input to our audio CNN and average the prediction scores to obtain the final prediction result. For evaluation on Charades and Kinetics-Sounds, we use three audio-visual sequences with different spatial crops from a test video and max-pool/average the prediction scores from each sequence, respectively.

Optimization. In all experiments, we use the AMSGrad [126] variant of AdamW [127] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, L2 weight decay of 1e-4. We use a learning rate warm-up for the first 6% of iterations followed by a linear decay of learning rate.

From the observations of Lezcano-Casado and Martínez-Rubio [123], we have 10 times less learning rate for the orthogonal parameters than that for the non-orthogonal parameters: we use 1e-5 for the former and 1e-4 for the latter.

We pretrain our model on Kinetics-700 with a batch size 256 for 220K iterations and AudioSet with a batch size 300 for 220K iterations in the main experiments; for the ablation study, we use a much smaller batch size of 4 and pretrain our model on Kinetics-700 for 80K iterations.

For finetuning on UCF101, we train our model for 40K iterations with a batch size of 64 and learning rate of 0.02. For evaluation on ESC-50, we train

a multi-class one-vs-all linear SVM on top of our fixed audio CNN for 38K iterations with a batch size of 128 and learning rate of 0.003. For finetuning on Charades, we train for 40K iterations with a batch size of 8, with learning rate of 0.001 for the classifier and CNN parameters, 1e-5 for the orthogonal parameters and 1e-4 for the rest parameters. For finetuning on Kinetics-Sounds, we train for 24K iterations with a batch size of 32, with learning rate of 0.005 for the classifier and CNN parameters, 1e-4 for the orthogonal parameters and 1e-3 for the rest parameters.

3.3.2 Results and Discussion

Multimodal Fusion Methods To evaluate different fusion methods on the quality of learned representation, we test the following settings: (i) `Early` uses a single multimodal Transformer with $2 \times L$ layers, (ii) `Mid` is our approach described in Figure 3.1, (iii) `Late` uses two unimodal Transformers each with $2 \times L$ layers. All the methods are pretrained on audio-visual data using CPP and MEP losses, except for (iv) `Late-w/o-CPP` where we use only the MEP loss. We finetune the pretrained models on audio-visual, audio-only, and visual-only scenarios. For fair comparisons across different fusion methods, we do not perform parameter sharing in this ablation setting.

Table 3.2 shows that `Early` and `Mid` outperform `Late` on the audio-visual scenario. This suggests the importance of encoding cross-modal information. Note that `Late-w/-CPP` gets cross-modal self-supervision, which gives marginal performance improvement over `Late-w/o-CPP`; however, both methods miss the opportunity to *encode* any cross-modal relationship, leading to inferior results. While both `Early` and `Late` perform similarly in the audio-visual scenario, only `Late` can be used in unimodal downstream scenarios (c.f., `Early` requires the presence of both modalities). This has practical implications: `Mid` and `Late`

Fusion Method	Audio-Visual	Audio-only	Visual-only
`Early`	64.9 / 89.8	- / -	- / -
`Late-w/-CPP`	61.0 / 88.7	52.3 / 80.8	41.0 / 71.3
`Late-w/o-CPP`	60.6 / 87.6	50.5 / 79.9	40.7 / 71.7
`Mid`†	**65.7** / **89.9**	**53.5** / **82.7**	**42.5** / **73.2**

Table 3.2: Ablation study on Kinetics-Sounds comparing multimodal fusion methods. We report top-1 and top-5 accuracy (%). †: Ours.

Sampling Method	top-1	top-5
`Current-Sequence`	64.6	89.8
`Current-MiniBatch`	65.5	90.8
`CANS-Dissimilar`	66.2	91.1
`CANS-Similar`†	**67.5**	**92.3**

Table 3.3: Ablation study on Kinetics-Sounds comparing negative sampling strategies. We report top-1 and top-5 accuracy (%). †: Ours.

Model	X.-L	X.-T	Params	top-1/5
`Multi-2`	✗	✗	7M	60.3 / 88.9
`Multi-6`	✓	✗	21M	65.7 / 89.9
`Multi-6`	✓	✓(`All`)	7M	67.1 / 92.3
`Multi-6`	✓	✓(`Part`†)	**4M**	**67.5** / **92.3**

Table 3.4: Ablation study on Kinetics-Sounds comparing parameter sharing schemes of Multimodal Transformers. X.-L: Cross-layer, X.-T: Cross-Transformer sharing. We report top-1 and top-5 accuracy (%). †: Ours.

Model	X.-L	X.-T	Params	top-1/5
`Vis-2`	✗	✗	14M	41.4 / 71.0
`Vis-2`	✓	✗	7M	41.2 / 72.9
`Vis-6`	✗	✗	43M	43.8 / 74.2
`Vis-6`	✓	✗	7M	43.5 / 73.7

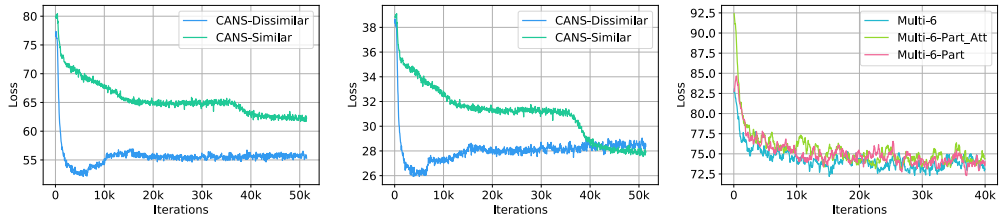
Table 3.5: Ablation study on Kinetics-Sounds comparing parameter sharing schemes of visual Transformers. X.-L: Cross-layer, X.-T: Cross-Transformer sharing. We report top-1 and top-5 accuracy (%). †: Ours.

can effectively handle missing modalities, i.e., once pretrained on audio-visual data, we can use it on any of audio-visual, audio-only, and visual-only scenarios. Our `Mid` fusion approach enjoys both the advantages, i.e., learning cross-modal relationship and being robust to missing modalities, achieving overall the best performance.

Negative Sampling Strategies We compare four negative sampling strategies: (i) `Current-Sequence` takes all but the positive instance from the same sequence as negatives, (ii) `Current-MiniBatch` takes all but the positive instance in the mini-batch as negatives; this subsumes `Current-Sequence`, (iii) `CANS-Dissimilar` stochastically samples negatives using a modified version of our content-aware negative sampling (CANS) that favors *dissimilar* samples, and (iv) `CANS-Similar` is our proposed CANS approach that favors negatives that are *similar* to the positive instance.

Table 3.3 shows `Current-Sequence` is the least effective: It makes MEP too difficult because negatives are (sometimes too much) similar to positives. As a result, the training dynamics is dominated by CPP, which is relatively easier, leading to inferior performance. We make quite the contrary observations from `Current-MiniBatch`: the inclusion of negatives from different videos makes MEP easier and thus makes it dominate the training dynamics. Our `CANS` approach solves both these issues by eliminating negatives that are either almost identical to or trivial to distinguish from the positives, based on the 95% CI over the CNN embedding distances. It also samples negatives in a stochastic manner so a wide variety of samples can be included as negatives. Our proposed `CANS-Similar` can be considered as a "softened" version of `Current-Sequence`; it samples negatives that are similar to positives with a high probability (this can be considered as online hard negative mining), but it also takes instances from different videos with a lower probability. This balances out hard and easy negatives, making the MEP task effective.

Figure 3.3 (a) provides additional evidence that supports our decision. We see that the loss of `CANS-Disimilar` initially drops rapidly but starts increasing around iteration 7K and continues to increase until around 15K; this is mainly caused by the visual MEP loss shown in Figure 3.3 (a-2). One explanation for



(a-1) MEP + CPP (a-2) Visual MEP

(a) Ablation: negative sampling

MEP + CPP

(b) Ablation: parameter sharing

Figure 3.3: Loss curves during pretraining under different ablative settings. **(a)** compares Content-Aware Negative Sampling (CANS) that favors negatives that are dissimilar vs. similar to the positive instance. **(b)** compares different cross-Transformer weight sharing schemes; see the text for details.

this might that `CANS-Disimilar` is easier to solve than `CANS-Similar`, which causes the loss landscape of `CANS-Disimilar` to contain too many shallow local minima compared to that of `CANS-Similar`. Recall that we use a learning rate warm-up for the first 6% of iterations during pretraining; this roughly equals to the first 13K (out of 220K) iterations. Given this, we speculate that the model got out of a local minima around iteration 7K (most likely due to the increasing learning rate), and then eventually settled in another (bad) local minima after the warm-up period ended. Compared to this, we observe much milder learning dynamics with `CANS-Similar`: the loss decreases relatively slowly but steadily, and eventually leaps around 35K to go below the loss of `CANS-Disimilar`. We, again, believe that this is because `CANS-Similar` is more difficult to solve than `CANS-Disimilar` (as shown by the slower decrease in loss values), which caused the resulting loss landscape to contain steeper local minima. Our model eventually found one of those after round 40K of iterations, resulting in a better performing model in the downstream tasks shown in Table 3.3 (the loss kept slowly decreasing after iteration 50K).

Parameter Sharing Schemes Our parameter reduction scheme reduces the number of parameters from 128M to 4M (by 97%) (Table 3.4. We reduce the model size by sharing weights across Transformers and across layers. We validate these ideas in two sets of experiments. Table 3.4 compares cross-Transformer weight sharing schemes. We use `Multi-6` that uses all three Transformers with 6 layers each, and compare four methods that correspond to Figure 3.2 (a)-(d). Note that `No sharing` is too large to fit in a Tesla V100 GPU (16GB) even with 2 samples, so we define `Multi-2` that uses three Transformers with 2 layers each, and with the reduced number of attention heads A to 5, the feature dimension D to 320 and the intermediate dimension E to 1280. We see that our proposed approach, `Part`, achieves the best performance with the least number of parameters. One might ask how `Part` leads to a smaller model when `All` shares all the weights across Transformers: We decompose weights $W = U\Sigma V^\top$ with low-rank approximation and share only U across Transformers, while the $\Sigma V^\top$ part learns modality-specific dynamics. Table 3.5 compares cross-layer weight sharing schemes using the visual Transformer with either 2 (`Vis-2`) or 6 (`Vis-6`) layers. The results show that sharing weights across layers does not hurt the performance, confirming the observations by Lan et al. [117] in the audio-visual setting.

Furthermore, recall that each layer of a Transformer contains the multi-head attention layer weights $\{W^q, W^k, W^v, W^b\}$ and the feed-forward layer weights $\{W^c, W^d\}$. We chose to share all six weight matrices across Transformers, though we could have shared any combination of them. To justify this design choice, we empirically compared three variants: (i) `Multi-6` that do not share parameters across Transformers, (ii) `Multi-6-Part_Att` that shares only $\{W^q, W^k, W^v, W^b\}$ (but not $\{W^c, W^d\}$) and (iii) `Multi-6-Part` that shares all six weight matrices. Figure 3.3 (b) shows that there is not much difference be-

tween all the variants in terms of the loss curves; we chose to use `Multi-6-Part` that requires the least number of parameters, and at the same time outperforms `Multi-6`.

Pretraining Objectives To evaluate the importance of MEP and CPP tasks, we test two settings: (i) `Mid-w/o-CPP` and (ii) `Mid-w/o-MEP`. On Kinetics-Sounds, these achieve 65.9% and 64.6%, respectively; ours achieve 67.5% (top-1 accuracy). The result show that the MEP task plays an important role during pretraining, confirming the findings from Sun et al. [115] that the InfoNCE loss, as deployed in CBT, is effective in the cross-modal setting. The result also shows that augmenting MEP with CPP provides further performance improvement by learning cross-modal correspondence.

Justification for the MEP Loss Formulation. Since the multimodal Transformer h_{AV} has access to both visual and audio inputs, one might think that the model could "leak" information about visual input into $\mathbf{z}^a$ and information about audio input into $\mathbf{z}^v$, which could make MEP trivial to solve. Here we show that this is not the case. By construction, we mask the same positions in audio and visual streams when designing the MEP task, so the model has no access to the masked input even in a cross-modal manner. Empirically, removing the third term in Equation 3.6 ($\mathcal{L}_{\text{NCE}}([\mathbf{x}^a; \mathbf{x}^v], \tilde{\mathbf{z}})$) leads to performance degradation in Kinetics-Sounds, i.e., top-1 accuracy 66.7% vs. ours 67.5%, which suggests that solving the MEP task in the multimodal Transformer is beneficial to our model.

Justification for the CPP Loss Formulation. Recall that our CPP loss has two terms; the first term uses the summary embeddings $\mathbf{s}_g$ and the second term uses output embeddings $\mathbf{s}_h$ sampled at random positions; see Equation 3.7. One could argue that the two terms are redundant as bidirectional Transformers

have "one-step" access to all the input embeddings, and thus solving CPP only with the summary embeddings (the first term) would be enough. This is not the case. We encode $\mathbf{s}_h$ with position-specific information through the time embeddings $\mathbf{p}_t$, which makes every $\mathbf{s}_h$ different compared to $\mathbf{s}_g$. Empirically, we find that removing the second term of Equation 3.7 ($\mathbf{s}_h$) in our CPP loss leads to an inferior accuracy 66.9% vs. ours 67.5% on Kinetics-Sounds, suggesting its importance in learning.

Use of Modality Embeddings in the Multimodal Transformer We use modality embeddings $\mathbf{m}^v$ and $\mathbf{m}^a$ as part of input to the multimodal Transformer in order to distinguish embeddings coming from visual and audio Transformers. They are learnable weights trained end-to-end with other parameters. Conceptually, incorporating modality-discriminative embeddings is crucial because of our aggressive weight sharing scheme. Without them, the multimodal Transformer will see the output from audio/visual Transformers (y^a and y^v) as if they are coming from the same distribution because the two Transformers share a large part of weights. Using modality embeddings encourages our model to preserve modality-specific information in the final output, and this empirically leads to performance improvements: ours 67.5% vs. without modality embeddings 67.1% on Kinetics-Sounds.

On the Importance of End-to-End Pretraining Previous work in multimodal visual-and-language tasks [112, 110] point out that using partially fixed Transformers of different modalities is detrimental to multimodal representation learning (c.f., [115, 61]). We make the same observation in our audio-visual learning scenario. We compare two variants of `Multi-6 Part` in Table 3.4, each of which pretrains only the audio (or visual) CNN/Transformer in the first half

Model	Net	Data	UCF
ST-Puzzle [35]	3D-R18	K400	65.8
ClipOrder [86]	R(2+1)D	UCF	72.4
DPC [87]	3D-R34	K400	75.7
CBT [115]	S3D	K600	79.5
MultiSens [39]	3D-R18	AS	82.1
AVTS [41]	MC3-18	K400	85.8
AVTS [41]	MC3-18	AS	**89.0**
`V-CNN`†	SlowFast	K700	85.2
`V-CNN`†	SlowFast	AS	86.1

Datasets. K: Kinetics, AS: AudioSet, UCF: UCF101.

Table 3.6: Short video classification results on UCF101 (UCF; mean accuracy (%)). †: Ours.

Model	Net	Data	ESC
SVM [2]	MLP	-	39.6
ConvAE [128]	CNN-4		39.9
RF [2]	MLP	-	44.3
ConvNet [129]	CNN-4	-	64.5
SoundNet [128]	CNN-8	FS	74.2
L^3-Net [3]	CNN-8	FS	79.3
DMC [130]	VGG-ish	FS	79.8
AVTS	VGG-M	AS	80.6
`A-CNN`†	R50	AS	**81.5**

Datasets. AS: AudioSet, FS: Flicker-SoundNet.

Table 3.7: Short audio classification results on ESC-50 (ESC; mean accuracy (%)). †: Ours.

Model	Charades	KS
Random	5.9	- / -
ATF [131]	18.3	- / -
ATF (OF) [131]	22.4	- / -
`V-CNN`	18.7	45.8 / 73.3
`A-CNN`	18.9	49.4 / 76.9
`M-CNN`	23.1	59.4 / 83.6
`V-BERT`	26.0	49.5 / 78.9
`A-BERT`	27.4	58.9 / 85.7
`M-BERT`†	**29.5**	**75.6** / **94.6**

Table 3.8: Long video classification results on Charades (mAP) and Kinetics-Sounds (KS; top-1/5 accuracy (%)). †: Ours.

of pretraining stage and then continues pretraining the remaining weights while fixing the weights of the audio (or visual) CNN/Transformer in the second half. This leads to inferior performance (audio-fixed 62.8% and visual-fixed 63.1% vs. ours 67.5%), which is consistent with the results reported in [112, 110].

Downstream Evaluation We pretrain our model with `Mid` fusion using MEP and CPP tasks (with `CANS-Similar`), and employ `Part` weight sharing. We use either Kinetics-700 or AudioSet for fair comparisons with prior work. Tables 3.6-3.7 show *short*-video/audio classification results on UCF101/ESC-50, respectively. For fair comparisons to the baselines, we use only the visual/audio CNN (no Transformers); we finetune a linear classifier on top of the visual CNN end-to-end for UCF101, and train a multi-class one-vs-all linear SVM on top of the fixed audio CNN for ESC-50. Although our model is pretrained on long video clips with no direct supervision to the CNN layers (gradients must flow *through* Transformers), it outperforms most of the baselines (except for AVTS on UCF101) that received direct supervision from short video clips. We note that, similar to ours, CBT [111] is a multimodal Transformer pretrained on long video clips and thus is the most meaningful comparison to ours; ours outperform CBT on UCF-101 by 5.7%. For sound classification, our approach outperform all existing published results.

Table 3.8 shows *long*-video classification results on Charades and Kinetics-Sounds (KS) when pretrained on Kinetics-700. We test `Visual-only (V)`, `Audio-only (A)`, and `Multimodal (M)` settings to verify the benefit of multimodal learning. Because there is no published self-supervised learning results on these datasets, we demonstrate long-term representations by comparing CNNs (`CNN`; short-term) to Transformers (`BERT`; long-term) on KS that contains 10-second clips. Since CNNs process 1-second clips, we feed 10 non-overlapping clips to CNNs and average the prediction output. In all settings, we add a 2-layer MLP with softmax classifier on top. The results show that Transformers outperform CNNs on Kinetics-Sounds, suggesting the superiority of long-term representations. We also see that combining audio-visual information performs the best. We notice that audio representations are generally stronger than visual representations;

we believe that learning discriminative visual representations is generally more challenging, especially when the CNNs receive (self-)supervision signals *only indirectly* through Transformers. We believe that providing (self-)supervision directly to CNNs, e.g., by first pretraining CNNs on 3D rotation prediction [37] and then jointly training the whole model (as was done in CBT [115]), could further improve performance. Incorporating contrastive learning [81] over the CNN embeddings and training the whole model end-to-end is another promising direction for future work.

3.4 Related Work

Multimodal BERT. Extending BERT [46] to vision-and-language has been actively studied. Existing work typically adopt early fusion [132, 133, 61, 134, 135, 111, 136, 137] or mid fusion [112, 110, 115, 138] without thorough validation, and they train only visual components while relying on a language-pretrained BERT. Although there have been some efforts to leverage the Transformer architecture [54] for audio and visual inputs [139, 140], our approach is the first to demonstrate multimodal audio-visual BERT trained from scratch in an end-to-end manner. This is enabled by our novel parameter reduction technique, which is one of our main technical contributions.

Audio-Visual Learning. Early work in audio-visual learning focused on speech signals, improving audio-visual speech recognition than unimodal approaches [141, 142]. Recent approaches leverage unlabeled videos from specific domains [143, 103, 144, 102, 104, 145, 40] and often demonstrate on audio-visual source separation, localization, and co-segmentation. However, these approaches rely on short-term audio-visual correspondence and thus may not generalize to long-term video recognition that requires global context (as was suggested in Hjelm et al. [95]), which this work focuses on.

Parameter Reduction. Network pruning [146, 147] trains a large model and then reduces its size while maintaining performance. Reducing the size of CNNs for mobile applications is an active research area [148, 149, 150, 151, 152]. Our work is closely related to the work that shares parameters across layers in deep neural networks. Trellis network [153] is a temporal convolutional architecture with weight-tying across time and depth. Similar to ours, Universal Transformer [154], RSNMT [155], DEQ [124], ALBERT [117] share weights across layers in Transformers. We combine this idea with our novel cross-Transformer weight sharing, which decomposes weight matrices with low-rank approximation.

Negative Sampling. Hard negative mining has been shown to be crucial for contrastive learning [3, 39, 41, 156, 157, 42, 158]. Korbar et al. [41] use the time difference between clips to approximate clip similarity (i.e., clips that are further apart are deemed more different). However, such an assumption may not hold for real-world videos, e.g., periodic actions such as push-ups. Unlike this line of approaches, we directly use the feature embeddings learned by our model. Several approaches adapted a similar idea [156, 157, 42, 158]. Different from prior work, we bring the stochasticity to the sampling procedure by using the content similarity as the sampling probability; this helps reduce potential errors especially during the early stage of training.

3.5 Conclusion

We introduced a multimodal bidirectional Transformer architecture for self-supervised learning of contextualized audio-visual representation from unlabeled videos. Our main technical contributions include: (1) we propose a parameter efficient multimodal Transformers based on matrix decomposition with low-rank approximation; (2) we propose a novel content-aware negative sam-

pling technique for contrastive learning. We demonstrate a successful end-to-end training of multimodal Transformers for audio-visual learning (which is, to the best of our knowledge, the first time in the literature). We also report comprehensive evaluation of various design decisions in multimodal learning.

Chapter 4

Massively Harvesting Good Video Data to Learn from without Human Efforts

4.1 Introduction

Our long-term objective is learning to recognize objects, actions, and sound in videos without the need for manual ground-truth labels. This is not only a theoretically interesting problem, since it mimics the development of auditory and visual perception by infants [159], it is also of immense practical importance, since accurate manual labeling of audio-visual data is impractical. Compared to self-supervised learning on static images [94, 95, 82, 81], audio-visual inputs pose additional challenges: large portions of a video may contain no relevant information, and auditory and visual inputs may not always be in correspondence. Consequently, existing self-supervised methods on audio-visual data either start with datasets for which there is a high probability of audio-visual correspondence, or they learn audio-visual properties corresponding only to short-term

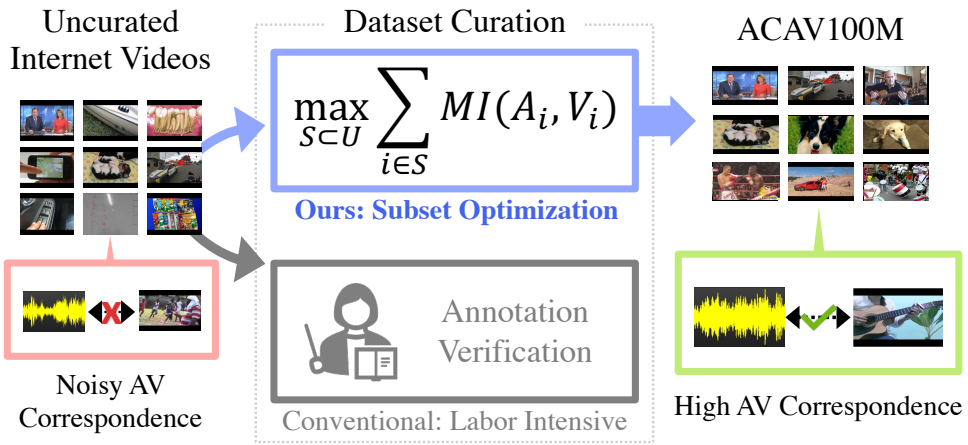


Figure 4.1: We address the challenge of constructing a large-scale audio-visual dataset from uncurated Internet videos without relying on manual annotation or verification. We solve a constrained optimization problem that finds a subset maximizing the mutual information between audio and visual signals in videos. The result is a new 100M video dataset with high audio-visual correspondence, ideal for self-supervised video representation learning.

statistical regularities. The necessary datasets are usually manually created or rely on domain-specific properties (e.g., [106, 5] and below). If we want to carry out self-supervised learning on full length (minutes, hours) of video without manually generating and/or selecting video clips, we need automated ways of curating such collections of audio/video clips from diverse collections of full length video.

We consider self-supervised learning from unlabeled videos as a two-step process: (1) an automatic dataset curation process that generates short, relevant clips with useful self-supervisory signals, e.g., audio-visual correspondence, and (2) a self-supervised learning approach that operates on the collection of short clips. This paper focuses on step (1) and *not* on step (2), providing an automated way of taking a collection of general or domain-specific videos of arbitrary

length and reducing it to a collection of shorter clips containing a high portion of relevant audio-video correspondences. The output of this step is a dataset, which can be used as input to existing self-supervised algorithms on audio-visual data [41, 104, 29], as well as the development of novel self-supervised techniques.

To achieve step (1), we assume access to a large collection of unconstrained videos and solve a subset selection problem with an information-theoretic measure of audio-visual correspondence as a selection criterion. Specifically, we find a subset that maximizes mutual information (MI) between audio and visual channels of videos. This is a necessary condition for self-supervised learning approaches that rely on audio-visual correspondence [160]. The main technical challenge we address is how to *efficiently* measure the audio-visual MI and find a subset that maximizes the MI in a scalable manner. Given that video processing is notoriously compute and storage intensive, we put a particular emphasis on scalability, i.e., we want an approach that can easily handle hundreds of millions of video clips.

MI estimation has a long history of research [161, 162], including the recent self-supervised approaches [94, 95, 81] that use noise contrastive estimation [116] as the learning objective. While it is tempting to use such approaches to estimate MI in our work, we quickly encounter the "chicken-and-egg" problem: to obtain such models for estimating audio-visual MI, we need a training dataset where we can reliably construct positive pairs with a high probability of audio-visual correspondence; but that is what we are set out to find in the first place! One might think that randomly chosen videos from the Internet could be sufficient, but this has shown to produce suboptimal representations [104]; our empirical results also show that self-supervised models indeed suffer from noisy real-world audio-visual correspondences.

In this work, we turn to a clustering-based solution that estimates the MI by measuring the agreement between two partitions of data [163, 164]. To circumvent the "chicken-and-egg" issue, we use off-the-shelf models as feature extractors and obtain multiple audio and visual clusters to estimate the MI. The use of off-the-shelf models is a standard practice in video dataset generation. Unlike existing approaches that use them as concept classifiers [165, 166, 167, 168, 4], here we use them as generic feature extractors. To avoid estimating the MI based on a restricted set of concepts the off-the-shelf models are trained on, we perform clustering over features computed across multiple layers (instead of just the penultimate layers), which has been shown to provide general feature descriptors not tied to specific concepts [169].

To make our approach scalable, we avoid using memory-heavy components such as the Lloyd's algorithm [170] and instead use SGD [171] to perform K-means clustering. Further, we approximately solve the subset maximization objective with a mini-batch greedy method [172]. Through controlled experiments with ground-truth and noisy real-world correspondences, we show that our clustering-based approach is more robust to the real-world correspondence patterns, leading to superior empirical performances than the contrastive MI estimation approaches.

We demonstrate our approach on a large collection of videos at an unprecedented scale: We process 140 million full-length videos (total duration 1,030 years) and produce a dataset of 100 million 10-second clips (31 years) with high audio-visual correspondence. We call this dataset ACAV100M (short for **a**utomatically **c**urated **a**udio-**v**isual dataset of **100M** videos). It is two orders of magnitude larger than the current largest video dataset used in the audio-visual learning literature, i.e., AudioSet [5] (8 months), and twice as large as the largest video dataset in the literature, i.e., HowTo100M [67] (15 years).

To evaluate the utility of our approach in self-supervised audio-visual representation learning, we produce datasets at varying scales and compare them with existing datasets of similar sizes that are frequently used in the audio-visual learning literature, i.e., Kinetics-Sounds [3] at 20K-scale, VGG-Sound [4] at 200K-scale, and AudioSet [5] at 2M-scale. Under the linear evaluation protocol with three downstream datasets, UCF101 [1], ESC-50 [2], and Kinetics-Sounds [3], we demonstrate that models pretrained on our datasets perform competitively or better than the ones pretrained on the baseline datasets, which were constructed with careful annotation or manual verification.

To summarize, our main contributions are: 1) We propose an information-theoretic subset optimization approach to finding a large-scale video dataset with a high portion of relevant audio-visual correspondences. 2) We evaluate different components of our pipeline via controlled experiments using both the ground-truth and the noisy real-world correspondence patterns. 3) We release ACAV100M, a large-scale open-domain dataset of 100M videos for future research in audio-visual representation learning.

4.2 Related Work

Large-Scale Data Curation. Several different types of audio-visual video datasets have been collected: (1) manually labeled, e.g., AudioSet [5], AVE [173], (2) domain specific, e.g., AVA ActiveSpeaker [174], AVA Speech [175], Greatest Hits [143], FAIR-Play [103], YouTube-ASMR-300K [176], and (3) unlabeled, unrestricted collections from consumer video sites, e.g., Flickr-SoundNet [128, 3].

AudioSet [5] contains about 2M clips corresponding to audio events retrieved from YouTube by keyword search; human raters verified the presence of audio events in the candidate videos. *Moments in Time* [177] contains over one mil-

lion clips of diverse visual and auditory events; video clips were selected using keywords (verbs) and manually reviewed for high correspondence between the clips and the keywords. *HowTo100M* [67] contains 136M clips segmented from 1.22M narrated instructional web videos retrieved by text search from YouTube, with an additional filtering step based on metadata. *Web Videos and Text* (WVT) [178] contains 70M clips obtained by searching the web with keywords based on the Kinetics-700 [106] categories and retaining both the video and the associated text. Chen et al. [4] created a dataset of 200K clips for audio-visual research; clips were originally obtained by keyword search on YouTube and frames were classified with pretrained visual classifiers. Since keywords and visual classes do not perfectly correspond, such correspondences needed to be manually reviewed and corrected on randomly sampled clips in an iterative and interactive process.

We are building systems for learning audio-visual correspondence on diverse, unrestricted inputs. This requires large amounts of training data, making manual collection and labeling costly and impractical. Unlike previous dataset curation processes that involve costly human intervention, we introduce an automatic and scalable data curation pipeline for large-scale audio-visual datasets.

Subset Selection. Our work focuses on data subset selection; extensive prior work exists in supervised [179, 180, 181, 182], unsupervised [183, 184], and active learning settings [185, 186]. Different criteria for subset selection have been explored in the literature. *Submodular functions* naturally model notions of information, diversity and coverage [187], and can be optimized efficiently using greedy algorithms [188, 189]. *Geometric criteria* like the coreset [190] aim to approximate geometric extent measures over a large dataset with a relatively small subset.

Mutual-information (MI) between input feature values and/or labels has

been used successfully [191, 192, 193] as a probablistically motivated criterion. We propose to use MI as an objective function for subset selection and make the following two unique contributions: First, we use MI to measure audio-visual correspondence within videos by formulating MI between the audio and visual features. Second, we apply MI for the large-scale video dataset curation problem. In case of clustering-based MI estimation, we demonstrate that optimizing MI objective with a greedy algorithm is a practical solution for building a large-scale pipeline.

4.3 Data Collection Pipeline

Our pipeline consists of four steps: (i) acquiring raw videos from the web and filtering them based on metadata, (ii) segmenting the videos into clips and extracting features with pretrained extractors, (iii) estimating mutual information (MI) between audio and visual representations, and (iv) selecting a subset of clips that maximizes the MI.

4.3.1 Obtaining Candidate Videos

We crawl YouTube to download videos with a wide variety of topics. Unlike previous work that use a carefully curated set of keywords [4], which could inadvertently introduce bias, we aim for capturing the natural distribution of topics present in the website. To ensure the diversity in topics, cultures and languages, we create combinations of search queries with diverse sets of keywords, locations, events, categories, etc., to obtain an initial video list.

Before downloading videos, we process the search results using metadata (provided by YouTube API) to filter out potentially low quality / low audio-visual correspondence videos. We use the duration to exclude videos shorter than 30 seconds (to avoid low quality videos) and longer than 600 seconds (to

avoid large storage costs). We also exclude videos that contain selected keywords (in either title or description) or from certain categories – i.e., gaming, animation, screencast, and music videos – because most videos exhibit non-natural scenes (computer graphics) and/or low audio-visual correspondence. Finally, we detect language from the titles and descriptions using fastText [194, 195] and keep the ones that constitute a cumulative ratio of 0.9, resulting in eight languages (English, Spanish, Portuguese, Russian, Japanese, French, German, and Korean).

The result is 140 million full-length videos with a total duration of 1,030 years (median: 198 seconds). To minimize the storage cost we download 360p resolution videos; this still consumes 1.8 petabytes of storage. Handling such large-scale data requires a carefully designed data pipeline. We discuss our modularized pipeline below.

4.3.2 Segmentation & Feature Extraction

Clip Segmentation. To avoid redundant clips, we extract up to three 10-second clips from each full-length video. We do this by detecting shot boundaries (using the `scdet` filter in FFmpeg) and computing pairwise clip similarities based on the MPEG-7 video signatures (using the `signature` filter in FFmpeg). We then select up to 3 clips that give the minimum total pairwise scores using local search [196]. This gives us about 300M clips.

Feature Extraction. To measure correspondence between audio and visual channels of the 300M clips, we need good feature representations. An ideal representation would capture a variety of important aspects from low-level details (e.g., texture and flow) to high-level concepts (e.g., semantic categories). However, such an oracle extractor is hard to obtain, and the sheer scale of data makes it impractical to learn optimal feature extractors end-to-end. Therefore, we use

the "off-the-shelf" pretrained models to extract features, i.e., SlowFast [11] pretrained on Kinetics-400 [71] and VGGish [197] pretrained on YouTube-8M [166] for visual and audio features, respectively.

4.3.3 Subset Selection via MI Maximization

Next, we select clips that exhibit strong correspondence between visual and audio channels. To this end, we estimate the mutual information (MI) between audio and visual signals. Computing the exact MI is infeasible because it requires estimating the joint distribution of high dimensional variables, but several approximate solutions do exist [198]. Here we implement and compare two approaches: a noise-contrastive estimator (NCE) [116], which measures MI in a continuous feature space, and a clustering-based estimator that computes MI in a discrete space via vector quantization. The former estimates MI for each video clip, while the latter estimates MI for a set of video clips. As we show later in our experiments, we find the clustering-based MI estimator to be more robust to real-world noise.

NCE-based MI Estimation

Contrastive approaches have become a popular way of estimating MI between different views of the data [94, 95]. We add linear projection heads over the precomputed audio/visual features and train them using the contrastive loss [81]. From a mini-batch $\{(v_i, a_i)\}_{i=1}^{N_b}$ where v_i and a_i are visual and audio features, respectively, we minimize

$$l(v_i, a_i) = -\log \frac{\exp(S(\mathbf{z}_i^v, \mathbf{z}_i^a)/\tau)}{\sum_{j=1}^{N_b} \exp(S(\mathbf{z}_i^v, \mathbf{z}_j^a)/\tau)}, \tag{4.1}$$

where $\mathbf{z}_i^v$ and $\mathbf{z}_i^a$ are embeddings from the linear projection heads, $S(\cdot, \cdot)$ measures the cosine similarity, and τ is a temperature term (we set $\tau = 0.1$). For each mini-batch we compute $l(v_i, a_i)$ and $l(a_i, v_i)$ to make the loss symmetric.

Once trained, we can directly use $S(\mathbf{z}^v, \mathbf{z}^a)$ to estimate audio-visual MI and find a subset by taking the top N candidates from a ranked list of video clips.

Clustering-based MI Estimation

MI Estimation. Clustering is one of the classical ways of estimating MI [163, 164]. Given two partitions of a dataset $\mathbf{X}$ w.r.t. audio and visual features, $\mathcal{A} = \{\mathbf{A}_1, \cdots, \mathbf{A}_{|\mathcal{A}|}\}$ and $\mathcal{V} = \{\mathbf{V}_1, \cdots, \mathbf{V}_{|\mathcal{V}|}\}$, we estimate their MI as:

$$\mathrm{MI}(\mathcal{A}, \mathcal{V}) = \sum_{i=1}^{|\mathcal{A}|} \sum_{j=1}^{|\mathcal{V}|} \frac{|\mathbf{A}_i \cap \mathbf{V}_j|}{|\mathbf{X}|} \log \frac{|\mathbf{X}||\mathbf{A}_i \cap \mathbf{V}_j|}{|\mathbf{A}_i||\mathbf{V}_j|}. \tag{4.2}$$

This formulation estimates MI in a discrete (vector-quantized) space induced by clustering, and thus the quality of clustering affects the quality of the estimator. A straightforward approach to obtaining $\mathcal{A}$ and $\mathcal{V}$ is to cluster videos using the output from the penultimate layers of the pretrained networks. However, this can introduce distributional bias specific to the datasets on which the networks are pretrained [169, 199]. To address this issue, we cluster samples over each output space induced by different layers of the networks. This allows the MI estimator to consider a wide range of abstract concepts, from low-level (such as textures) to high-level (such as object parts) [200].

Specifically, we use the feature spaces induced by the five convolutional blocks from each of the SlowFast and VGGish feature extractors. We then compute the average MI between *all pairs* of clusterings as our MI estimator. Let $\mathcal{CV}_{\mathbf{X}}^{(i)} = \{\mathbf{V}_1^{(i)}, \cdots, \mathbf{V}_{n_i}^{(i)}\}$ and $\mathcal{CA}_{\mathbf{X}}^{(i)} = \{\mathbf{A}_1^{(i)}, \cdots, \mathbf{A}_{m_i}^{(i)}\}$ denote the clustering results induced by the i-th convolutional block of the visual and audio feature extractors, respectively. We compute:

$$F(\mathbf{X}) = \sum_{(\mathcal{X}, \mathcal{Y}) \in \mathcal{C}_{\mathbf{X}}} \frac{\mathrm{MI}(\mathcal{X}, \mathcal{Y})}{{}_{10}C_2}, \tag{4.3}$$

where $\mathcal{C}_{\mathbf{X}}$ denotes the combination of two elements from $\{\mathcal{CV}_{\mathbf{X}}^{(i)}\}_{i=1}^{5} \cup \{\mathcal{CA}_{\mathbf{X}}^{(j)}\}_{j=1}^{5}$ and ${}_{10}C_2$ denotes the number of 2-combinations out of 10 elements, which equals to 45. This computes MI between layers from both within and across the extractors of different modalities (referred to as *combination* pairing scheme in Section 4.4.2).

Algorithm 2: Greedy Algorithm

Input: initial dataset $\mathbf{D}$, clustering-based MI estimator F, target subset size M
Output: $\mathbf{X} \subseteq \mathbf{D}, |\mathbf{X}| = M$
$\mathbf{X}_0 \leftarrow \emptyset$
for $i = 0$ **to** $M - 1$ **do**
  $x \leftarrow \text{argmax}_{x \in \mathbf{D} \backslash \mathbf{X}_i} F(\mathbf{X}_i \cup \{x\})$
  $\mathbf{X}_{i+1} \leftarrow \mathbf{X}_i \cup \{x\}$
end
$\mathbf{X} \leftarrow \mathbf{X}_M$
Return X

Algorithm 3: Batch Greedy Subset Selection

Input: initial dataset $\mathbf{D}$, MI estimator F, target subset size M, batch size b, selection size s
Output: $\mathbf{X} \subseteq \mathbf{D}, |\mathbf{X}| = M$
$\mathbf{X}_0 \leftarrow \emptyset, i \leftarrow 0$
while $|X_i| < M$ **do**
  Randomly sample $\mathbf{B} \subseteq \mathbf{D} \backslash \mathbf{X}_i, |\mathbf{B}| = b$
  $\mathbf{Y}_0 \leftarrow \emptyset, j \leftarrow 0$
  while $j < s$ **do**
    $x \leftarrow \text{argmax}_{x \in \mathbf{B} \backslash \mathbf{Y}_j} F(\mathbf{X}_i \cup \mathbf{Y}_j \cup \{x\})$
    $\mathbf{Y}_{j+1} \leftarrow \mathbf{Y}_j \cup \{x\}, j \leftarrow j + 1$
    if $|\mathbf{X}_i \cup \mathbf{Y}_j| = M$ **then** break
  end
  $\mathbf{X}_{i+1} \leftarrow \mathbf{X}_i \cup \mathbf{Y}_j, i \leftarrow i + 1$
end
$\mathbf{X} \leftarrow \mathbf{X}_i$
Return X

Batch Greedy Subset Selection. Since the MI estimator $F(\cdot)$ is a func-

tion of $\mathbf{X}$, we can formulate an optimization problem where the goal is to find a subset $\mathbf{X}$ that maximizes $F(\mathbf{X})$. In general, finding a global solution to problems such as ours is NP-hard and thus greedy heuristic solutions are used instead [201]. However, as shown in Algorithm 2, the greedy algorithm selects one sample in each iteration and re-evaluates the MI estimator $F(\cdot)$ on all the remaining candidates. This introduces a challenge to our setting because the time complexity is quadratic to the size of the population; this is clearly not scalable to 300 million instances.

Therefore, we approximate the typical greedy solution using the batch greedy algorithm [172], as shown in Algorithm 3. It randomly samples a batch $\mathbf{B}$ from the remaining pool of candidates, and searches for the next element to be included in the active solution set only within $\mathbf{B}$. This batch trick reduces the time complexity down to linear, i.e., $O(N \times |\mathbf{B}|)$, where N is the size of the input dataset. We demonstrate the efficacy of the algorithm in Section 4.4.

Stochastic Clustering. One missing piece in this pipeline is an *efficient* clustering algorithm scalable to hundreds of millions of instances. The most popular choice among various clustering methods is K-means clustering [202], which is a special case of mixture density estimation for isotropic normal and other densities. Typically, an expectation-maximization (EM) algorithm, such as Lloyd's [170], is used to find the cluster centers. Such algorithms require repeated computation of the distances of all samples from all k cluster centers, followed by cluster assignment, until convergence. Lloyd's algorithm updates cluster centers only after each pass through the entire dataset. But for very large datasets (like ours), a small subset usually contains enough information to obtain good estimates of the cluster centers, meaning that EM-style algorithms tend to take (perhaps too) many epochs to converge.

There are different strategies for addressing this issue, including random

sampling and subsetting, but a straightforward approach is to replace EM algorithm with an SGD [203, 171, 204]. In such an approach, for large datasets, convergence rate and final accuracy of the cluster centers are determined not by the total dataset size, but by the learning rate schedule. A straightforward SGD update rule is to compute the nearest cluster centers for each sample in a batch and then update the cluster centers using a convex combination of the cluster centers and their nearest samples, weighting the samples with a learning rate λ and the cluster centers with $(1-\lambda)$. However, mixture density estimators in general suffer from the problem that adding mixture components with zero probability does not change the mixture density; in practice, this means EM and SGD-based algorithms may end up with cluster centers that stop receiving updates at some point during the optimization.

We address this problem by estimating the mixture component utilization rate as the ratio of the total number of updates to the cluster center divided by the total number of estimation steps, and reinitializing cluster centers when that probability falls below $(1/k)^2$. In Section 4.4.2, we demonstrate that our mini-batch SGD update shows comparable accuracy to batch update in correspondence retrieval tasks.

4.4 Evaluation on Correspondence Retrieval

We systematically evaluate different components of our pipeline with synthetic correspondence-retrieval tasks, where we generate corresponding (positive) and non-corresponding (negative) pairs using CIFAR-10 [205], MNIST [206] and FSDD [207]. In each correspondence retrieval task, the goal is to discover the known corresponding samples among the non-corresponding pairs. To show the generality of the findings, we also experiment with Kinetics-Sounds [3] which exhibit real-world audio-visual correspondence.

4.4.1 Experimental Setting

Datasets We construct five datasets where each instance is a pair of samples with different correspondence types.

1/2) CIFAR10-Rotation/Flip. We use images from five randomly selected categories to construct a "positive pair" set, and use the rest for a "negative pair" set. For the positive set, we create pairs of images by sampling two different images from the same category (e.g., two images of a bird), and apply a geometric transformation to one of them; we apply either a 90° CCW rotation (CIFAR10-Rotation) or a horizontal flip (CIFAR10-Flip). The negative set follows the same process but each pair contains images from different categories. We categorize this type of correspondence as "Natural Class Correspondence" because pairings are made over natural semantic categories.

3/4) MNIST-CIFAR10/FSDD. We use images from five digit categories to construct a positive set and use the rest for a negative set. Different from above, correspondence is defined via an arbitrary class-level mapping, e.g., "digit 0" images map to the "car" images in CIFAR-10 or "digit 0" audio samples in FSDD. We take samples from the same categories to construct the positive set and samples from different categories for the negative set. We call these "Arbitrary Class Correspondence" to differentiate from above.

5) Kinetics-Sounds. Unlike the above datasets where the correspondence is defined over class categories, here the correspondence is defined at the *sample* level, i.e., a positive set contains pairs of audio and visual channels of the same video, and a negative set contains randomly permuted pairs. We do not utilize class labels to construct the dataset.

Methods We compare our pipeline (both contrastive-based and clustering-based) to three ranking-based approaches. All the methods use the same pre-

computed features. For images, we use ResNet-50 [74] pretrained on ImageNet [208]. For videos, we use SlowFast [11] pretrained on Kinetics-400 [71] and VGGish [197] pretrained on YouTube-8M [166] for visual and audio features, respectively. For the ranking baselines, we apply PCA [209] to reduce the feature dimensionality to 64 and rank the instances based on three similarity metrics: inner product, cosine similarity, and (negative) l_2 distance. Because all our datasets have an equal number of positive and negative instances, we simply select the top 50% instances as the retrieval result.

Protocol We split each dataset into train and test partitions of the same size. We conduct a total of five runs for each of the five datasets and report results on the test splits. We use train sets only for the contrastive estimator to train the projection heads. When constructing each dataset, we sample at most $n = 1000$ instances from each category of the source datasets. For the noise contrastive estimator, we train the linear projection heads for 100 epochs using the AMSGrad variant of Adam optimizer [126] with a learning rate of 2e-4. We randomly take one sample from each class to build a mini-batch for class-level correspondence datasets, and sample random $N_b = 10$ clips to build a mini-batch for the sample-level correspondence dataset. When applying our clustering-based method, we perform the SGD K-means clustering with the "ground-truth" number of centroids as the number of classes in each source dataset; we use the batch greedy algorithm with a batch size $b = 100$ and a selection size $s = 25$.

4.4.2 Ablation Results & Discussion

Tables 4.1-4.3 show that the two variants of our approach – contrastive and clustering – achieve overall higher precision rates than the ranking baselines. As

Method	CIFAR10-Rotation	CIFAR10-Flip
Ranking-inner	87.872 ± 0.002	87.044 ± 0.001
Ranking-cos	87.872 ± 0.002	87.044 ± 0.001
Ranking-l_2	87.872 ± 0.002	87.044 ± 0.001
Ours-Contrastive	**99.395** ± 0.000	**99.480** ± 0.001
Ours-Clustering	87.292 ± 0.014	87.248 ± 0.010

Table 4.1: Results of natural class correspondence retrieval on CIFAR10-Rotation and CIFAR10-Flip. We conduct a total of five runs and report the precision with the 99% confidence interval. We use the clustering pairing scheme which gives the highest score in each configuration: combination, except diagonal for Ranking-inner, Ranking-cos and Rank-l_2.

Method	MNIST-CIFAR10	MNIST-FSDD
Ranking-inner	63.076 ± 0.001	64.453 ± 0.003
Ranking-cos	67.600 ± 0.002	61.893 ± 0.004
Ranking-l_2	66.796 ± 0.001	62.933 ± 0.003
Ours-Contrastive	73.252 ± 0.040	**73.733** ± 0.027
Ours-Clustering	**77.224** ± 0.009	69.440 ± 0.049

Table 4.2: Results of arbitrary class correspondence retrieval on MNIST-CIFAR10 and MNIST-FSDD. We conduct a total of five runs and report the precision with the 99% confidence interval. We use the clustering pairing scheme which gives the highest score in each configuration: combination.

Method	Kinetics-Sounds
Ranking-inner	52.558 ± 0.002
Ranking-cos	60.108 ± 0.001
Ranking-l_2	51.236 ± 0.001
Ours-Contrastive	73.066 ± 0.036
Ours-Clustering	**88.705** ± 0.004

Table 4.3: Results of audio-visual correspondence retrieval on Kinetics-Sounds. We conduct a total of five runs and report the precision with the 99% confidence interval. We use the clustering pairing scheme which gives the highest score in each configuration: combination.

Layers	Method	Precision
Single	Layer1	50.820 ± 0.014
	Layer2	51.412 ± 0.011
	Layer3	52.659 ± 0.012
	Layer4	54.422 ± 0.012
	Layer5	58.418 ± 0.030
Multiple	Diagonal	71.450 ± 0.005
	Bipartite	76.969 ± 0.005
	Combination	**88.705** ± 0.004

Table 4.4: Correspondence retrieval results on Kinetics-Sounds with different clustering pairing schemes. We conduct a total of five runs and report the precision with the 99% confidence interval.

shown in Table 4.1, the contrastive approach performs well on the two datasets with the "natural class correspondence," conforming to the previous results that shows contrastive learning is robust to geometric transformations [81]. Table 4.3 especially shows that the clustering approach excels on Kinetics-Sounds that contains natural audio-visual correspondence, which is closer to our intended scenario. Therefore, we conduct various ablation studies on Kinetics-Sounds to validate different components of our clustering-based approach.

Multi-Layer Clustering. All the feature extractors that we use consist of five convolutional blocks. As discussed in Section 4.3.3, we cluster samples over each of the five output spaces to capture a wide range of abstract concepts. This raises a question: How should we combine audio-visual clusters for MI estimation? Table 4.4 compares the single-layer approaches to multi-layer approaches. Each of the single-layer approach estimates the audio-visual MI based on a single pair of clustering results. We can see that the precision increases as we use clustering results from higher layers. However, all single-layer methods perform significantly worse than multi-layer variants.

We explore three options to select pairs of clusterings for MI estimation. `Diagonal` computes an average MI across all five single-layer scores (with L

Method	Layer Weights					Precision
	1	2	3	4	5	
exp(-10)	5e+10	2e+04	1	5e-05	2e-09	50.791
exp(-1)	7.4	2.7	1	0.4	0.1	65.374
exp(1)	0.1	0.4	1	2.7	7.4	79.858
exp(10)	2e-09	5e-05	1	2e+04	5e+10	57.880
linear(-0.50)	1.9	1.5	1	0.5	0.1	88.018
linear(-0.25)	1.5	1.2	1	0.8	0.5	88.673
linear(0.25)	0.5	0.8	1	1.2	1.5	88.777
linear(0.50)	0.1	0.5	1	1.5	1.9	87.997
Uniform (Ours)	1	1	1	1	1	88.705

Table 4.5: Different layer weighting schemes in clustering-based MI estimation using Kinetics-Sounds with `Combination` pairing.

layers, this computes MI L times), `Bipartite` computes an average MI between all possible combinations of audio-visual clustering results (L^2 times), and `Combination` (**ours**) computes an average MI between all possible combinations of clustering results, regardless of modalities (${}_{2L}C_2$ times). We observe that the performance increases with the number of connections as shown in the bottom rows of Table 4.4. This positive relationship suggests that the consensus between layers from the same extractor, as well as that across extractors, contributes to the clarity of correspondence signal.

Table 4.5 compares different layer weighting schemes for the `Combination` approach, which shows that our multi-layer approach is generally robust to weight distributions. We explore two alternative weighting schemes: a *linear(k)* function with slope k and an *exp(k)* function with slope e^k. We can see that precision is stable under a linear weighting scheme while precision drops significantly when the weights have a steep slope (e.g., *exp(-10)*), which is a degenerate case similar to the single-layer approach reported in Table 4.4. By default, we use *uniform* weights.

Mini-Batch SGD K-means Clustering. We compared mini-batch SGD

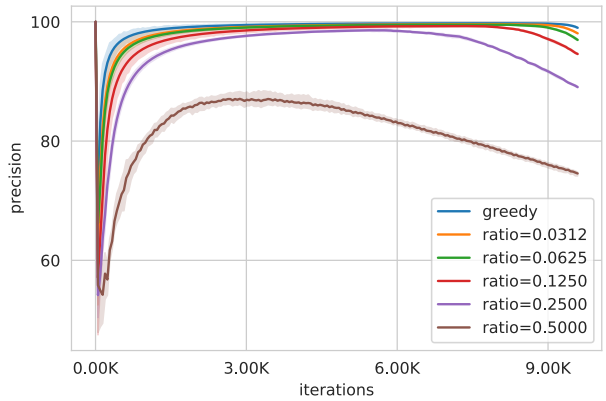


Figure 4.2: **Greedy vs. batch greedy algorithms** with varying selection-to-batch size ratios, s/b. The shaded regions show 99% confidence intervals obtained by five runs on Kinetics-Sounds. The batch greedy algorithm is robust when the ratio is $\leqslant 25\%$.

K-means [171] to the standard EM (Lloyd's) approach [170] and obtained very similar results on Kinetics-Sounds: 88.705 ± 0.004 (SGD) versus 88.732 ± 0.005 (EM). This shows that our SGD solution has negligible performance degradation while enjoying a significantly less memory requirement than the standard EM approach.

Batch Greedy Subset Selection. We explore how the use of mini-batches affects the quality of the selected subsets. We compare the greedy algorithm and the batch greedy algorithm with a batch size $b = 160$ and varying selection sizes $s = \{5, 10, 20, 40, 80\}$. As shown in Figure 4.2, the performance gap between the greedy algorithm and the batch greedy algorithm is marginal (greedy: 98.970 vs. batch greedy with $(b, s) = (160, 5)$: 98.020), which validates our use of the batch greedy algorithm. While the batch size itself does not have a large impact on the subset quality, the ratio of selection size to batch size (s/b) highly affects the retrieval performance; the performance drops sharply as the ratio exceeds 0.25 in several (b, s) configurations. This is mainly dataset-dependent: by construction,

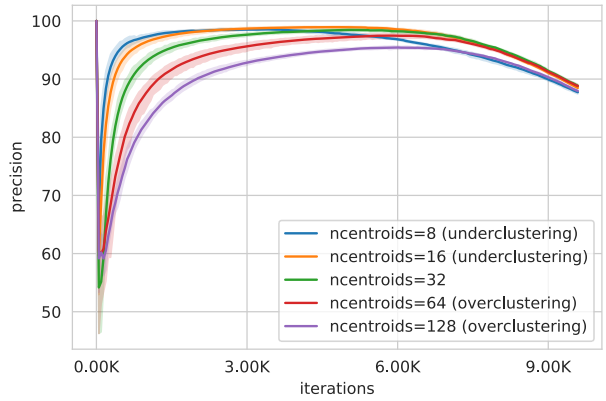


Figure 4.3: **Sensitivity analysis on the number of centroids**. We determine under/over-clustering based on the ground-truth number of class categories in Kinetics-Sounds ($c = 32$). The shaded regions show 99% confidence intervals over five runs.

there is a 50% chance that a sample will be a positive. We believe that the constructed dataset contains roughly 25% *easy positives*, i.e., videos with very high correspondence. When the selection ratio s/b does not exceed the easy positive ratio, the batch greedy algorithm finds those videos without introducing false positives, providing robustness. We found similar patterns with other ratios of $s/b > 25\%$.

Number of Centroids. We vary the number of centroids $k \in \{8, 16, 32, 64, 128\}$ to see how sensitive our approach is to the parameter. We apply the batch greedy algorithm with a batch size $b = 100$ and a selection size $s = 25$ on Kinetics-Sounds. Figure 4.3 shows that, although the final performance is similar across different number of centroids, they show different trends: underclustering ($k = \{8, 16\}$) shows high precision in early iterations while overclustering ($k = \{64, 128\}$) shows slower drop in the later stage.

4.5 Large-Scale Evaluation

We construct datasets at varying scales (20K, 200K, 2M) and compare them to existing datasets often used in the audio-visual learning literature: Kinetics-Sounds [3] (20K), VGG-Sound [4] (200K), and AudioSet [5] (2M). Note that all three datasets involve either human annotation [3, 5] or manual verification [4]. To demonstrate the scalable nature of our approach, we also generate datasets with 10M and 100M videos and evaluate their performance.

4.5.1 Automatic Data Curation

NCE-Based MI Estimation. For the contrastive approach, we train linear projection heads using Equation 4.1 with a batch size of $N_b = 1024$ from a randomly drawn set of 100M videos. Note that these additional videos are only used to train projection heads for MI estimation, which is discarded once dataset curation is finished; all approaches use the same number of videos under the same evaluation protocol on all downstream tasks. We train the model for three epochs and rank the entire video set (300M) based on the cosine similarity [81]. We then take top $N \in \{20\text{K}, 200\text{K}, 2\text{M}\}$ ranked videos for the final dataset.

Clustering-Based MI Estimation. For SGD K-Means clustering, we train the cluster centroids with a mini-batch of size 100K for 100 epochs using a learning rate $\lambda =$ 1e-2. When applying the batch greedy algorithm, we use the fixed batch size $b = 10000$ and the selection size $s = 500$ (with a ratio of $s/b = 0.05$), but vary the number of clusters $C \in \{100, 200, 500, 1000, 2000\}$ for each size of the datasets.

4.5.2 Linear Evaluation on Downstream Tasks

To assess the quality of the datasets, we pretrain identical models on different datasets and evaluate their performance on downstream tasks. The idea is that

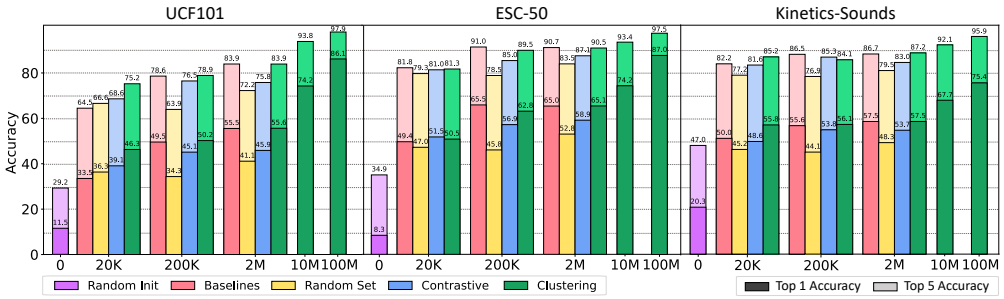


Figure 4.4: **Linear evaluation on downstream tasks**. The top-1/5 accuracy (%) of video classification on UCF101 [1], audio classification on ESC-50 [2] and audio-visual classification on Kinetics-Sounds (KS) [3]. We group the results by the downstream tasks and by the scale of the pretrain datasets. Baselines are Kinetics-Sounds [3] (20K), VGG-Sound [4] (200K), and AudioSet [5] (2M).

if a model performed particularly better than the others, the dataset used to train that model must be superior to the other datasets. We pretrain audio-visual CNNs from scratch using the self-supervised objective of SimCLR [81]; we use 3D ResNet-50 [210] and ResNet-50 [74] as the visual and audio CNNs, respectively. We follow the linear evaluation protocol [81] by adding a linear classifier on top of the learned and frozen models. We test on three downstream tasks: visual action recognition on UCF101 [1], sound classification on ESC-50 [2], and audio-visual action recognition on Kinetics-Sounds [3] (we concatenate audio-visual features for the linear classifier). Note that the training procedures are identical for all the models except for the datasets used to train them. We report mean accuracy across the official splits of UCF101 and ESC-50.

Figure 4.4 shows that models pretrained on our dataset (green bars) achieve similar, or even slightly better, performances compared to the baseline datasets (pink bars) at 20K, 200K, and 2M scales. The significant gap between ours vs. random set (yellow bars) shows the improvement does not come from the initial pool we crawl (the 300M set) but rather come from higher portion of

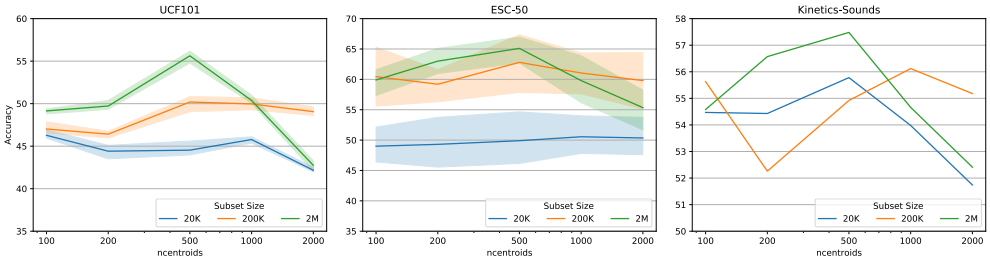


Figure 4.5: Linear evaluation of representations pretrained on the datasets that are constructed by our clustering-based approach. We report the top-1 accuracy (%) on UCF101 [1], ESC-50 [2], and Kinetics-Sounds [3], grouped by the number of cluster centroids. The shaded regions show 99% confidence intervals obtained by runs over the official splits of UCF101 (3 splits) and ESC-50 (5 splits).

audio-visual correspondence in the resulting dataset. Our clustering approach to MI estimation (green bars) generally outperforms the contrastive approach (blue bars), suggesting its robustness to noisy real-world audio-visual correspondences. Finally, we report the results obtained from 10M and 100M datasets produced with our clustering-based MI estimation module (we omit the baseline results at these scales due to computational reasons). The significant performance boost from the 10M and 100M models reaffirms the importance of large-scale training. Considering our data curation process does not involve human intervention (i.e., no manual annotation and verification), this is a promising result showing the potential for large-scale self-supervised learning: one can obtain datasets of arbitrary scales and develop self-supervised models by leveraging high portion of audio-visual correspondences provided in the datasets.

Impact of the Number of Centroids. To visualize the impact of the number of clusters in our clustering-based approach, we group the results by the number of clusters as shown in Figure 4.5. Notice that the number of clusters is not positively correlated with downstream task performance. Instead, clustering with about 500 clusters seems to yield the best performance. Fur-

Dataset	Majority Vote (%)	Fleiss' Kappa
AudioSet	65.66	0.4385
VGG-Sound	84.00	0.4634
Ours (2M)	69.00	0.5110
Random	44.00	0.6112

Table 4.6: **Human evaluation results** assessing the perceived audio-visual correspondence in videos from different datasets.

thermore, experiments using the largest number of centroids ($C = 2000$) show low accuracy consistently across all datasets and subset sizes. This confirms our findings in Section 4.4.2: overclustering tends to have a negative impact on the quality of the selected subset. We believe that this happens because, as the number of clusters increases, samples with homogeneous concepts in large clusters are scattered into small clusters sharing similar concepts. When we do not have many references to compare as in the early stage of subset selection, this fragmentation effect inhibits sample count sharing between conceptually similar small clusters, complicating the clustering-based MI estimation.

4.5.3 Human Evaluation

We conduct a user study to assess the perceived presence/absence of audio-visual correspondence in video clips. We compare clips from four datasets: AudioSet [5], VGG-Sound [4], ours with clustering (2M scale, 1K clusters), and random (drawn from the 300M set). We prepare 100 randomly sampled clips from each of these datasets, for a total of 400 clips. We recruit 12 participants and present each with 100 clips (25 clips per dataset), and ask them whether audio and visual are corresponding or not. This provides us with 3 votes per video.

Table 4.6 shows the majority voting accuracy and inter-rater agreement (measured by Fleiss' Kappa [211]). Every dataset has Fleiss' Kappa greater than

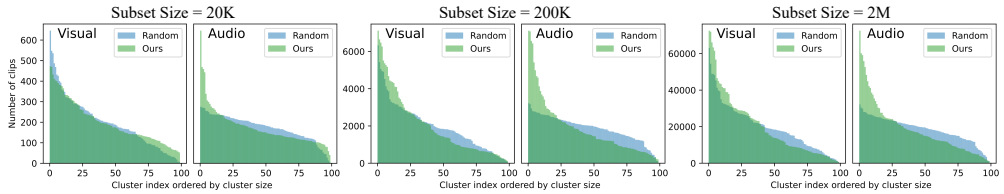


Figure 4.6: Histograms of cluster IDs from our curated subsets and randomly sampled subsets (with 100 cluster centroids). The blue histograms represent the case where samples are drawn uniformly random and thus is the unbiased representation of the concepts naturally appearing in the entire population.

0.4, verifying the reliability of the accuracy statistics [212]. Ours significantly improves audio-visual correspondence over a random subset (69% vs. 44%), and is even rated slightly higher than AudioSet. The annotation process for AudioSet has focused on audio events so we suspect that several of videos do not contain *visible* sound sources. There is still a significant gap between ours and VGG-Sound; we note that our process finds audio-visual correspondence without relying on manual verification as was done in VGG-Sound.

4.5.4 Diversity of the Sampled Clips

Histogram of Cluster IDs. To analyze the diversity of concepts contained in our curated dataset, we examine the histograms of cluster IDs from the chosen videos. Figure 4.6 shows audio and visual histograms obtained from either our curated subsets or randomly sampled subsets at varying scales (20K, 200K, and 2M). To obtain these, we cluster the features from the last layer of audio and visual feature extractors, respectively, and plot the histograms of cluster IDs. For the purpose of visualization we sort the cluster indices by the cluster size in a decreasing order (and thus the cluster IDs do not match between "Random" and "Ours" in each of the plots). The histograms from random subsets represent the natural distribution of the entire video population.

In the visual domain, the curated datasets (green histograms) mostly follow the original cluster distributions (which is reflected in the blue histogram in each subplot). This indicates that the visual concept distribution largely follows the natural distribution in the entire population, suggesting that our subset contains visual concepts that are as diverse as the entire set.

On the other hand, the audio clusters show noticeable concentration in distribution after subset selection. Upon close inspection of videos from the largest audio clusters, we observe that our curated datasets tend to choose videos from clusters with high audio-visual correspondence (e.g., videos of a single person speaking with no other sound in background) while random sampling tend to choose videos from clusters with no apparent audio-visual correspondence (e.g., videos of multiple people taking with background music/noise). This shows that the concentration in the audio histograms is caused by filtering out videos of low audio-visual correspondence, which is a highly desirable artifact in the curated subset.

Qualitative Analysis of Audio-Visual Clustering Results. To further investigate the diversity of concepts appearing in our subsets, we manually inspect audio and visual clustering results in the 2M dataset and compare the concepts appearing in the largest clusters to those in the smallest ones. Figure 4.7 and Figure 4.8 show representative videos from the five largest and five smallest clusters obtained from audio and visual clustering results, respectively. Figure 4.7 (from audio clusters) suggests that our curated dataset contains diverse concepts including general sound categories (e.g., voice and objects sounds) as well as specific topics (e.g., outdoor interview and cooking). Similarly, Figure 4.8 (from visual clusters) also suggests that our dataset contains diverse concepts including both natural (e.g., animals and fire) and human sounds (e.g., makeup and playing guitar). Clips from larger clusters (depicted in the left col-

umn of Figure 4.7 and Figure 4.8) contain clear and isolated sound sources, while sounds of smaller clusters (the right column) are less distinguishable due to multiple sound sources or background noise. Our dataset also captures several audio-visual concepts that existing datasets (such as VGG-Sound [4] and AudioSet [5]) do not offer. For instance, in Figure 4.7, the 77th cluster contains videos recorded from a front-facing camera with voice recordings from a phone mic, and the 46th cluster contains videos of comedians performing exaggerated body actions with the sound of crowd (cheering and laughter). The 88th cluster in Figure 4.8 contains shoes unboxing videos.

4.6 Conclusion

This work complements existing line of research on self-supervised representation learning with three main contributions: i) proposing an automatic and scalable data collection pipeline for audio-visual representation learning, ii) demonstrating that the MI-based subset selection can retrieve correspondence in both artificial and practical settings, and iii) releasing a large-scale open-domain video dataset consisting of 100M clips curated with our pipeline.

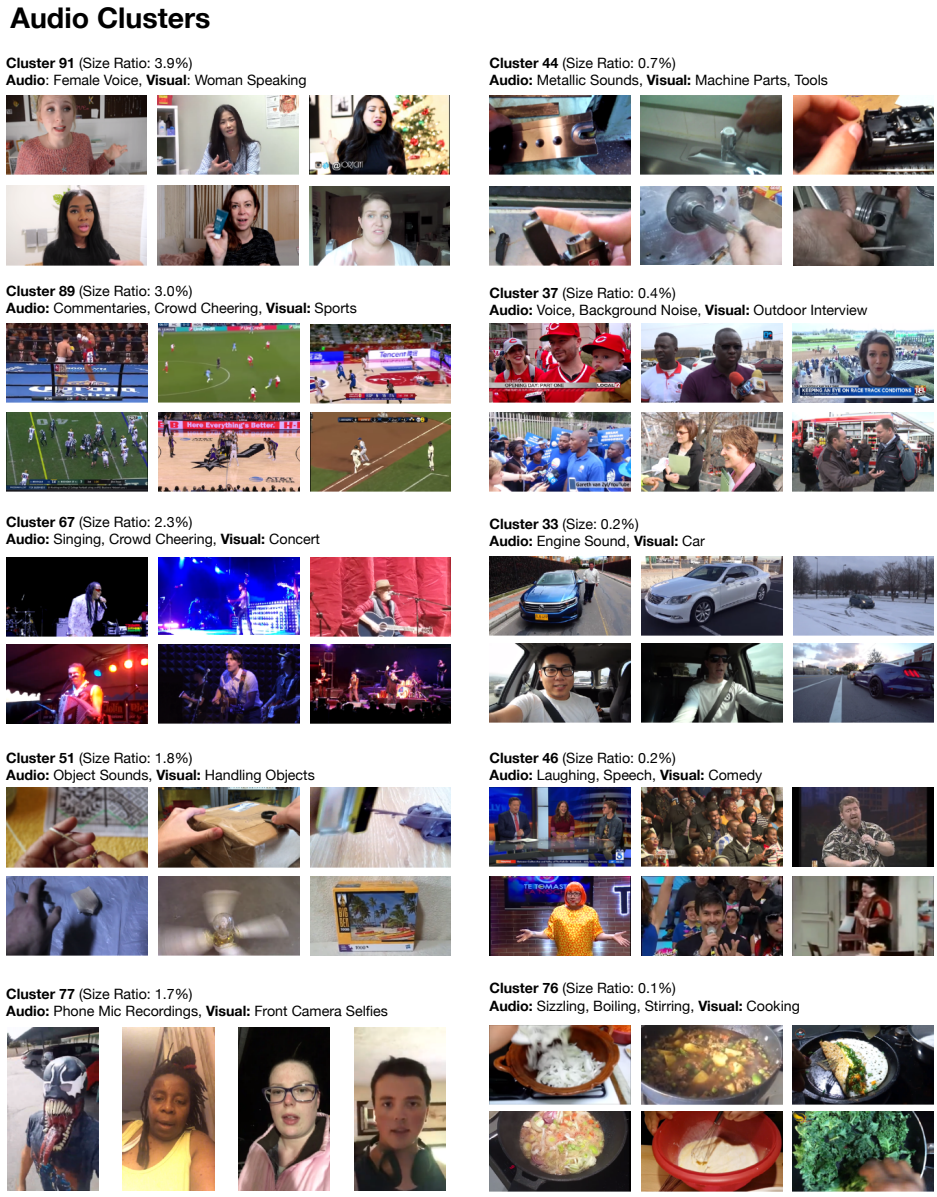


Figure 4.7: Representative samples and concepts derived from a manual inspection of 100 *audio* clusters of the 2M subset. We show samples from the five largest clusters on the left column and those from the five smallest clusters on the right. Each cluster captures distinctive audio-visual concepts, indicating that our curated subset contains various concepts with high audio-visual correspondence.

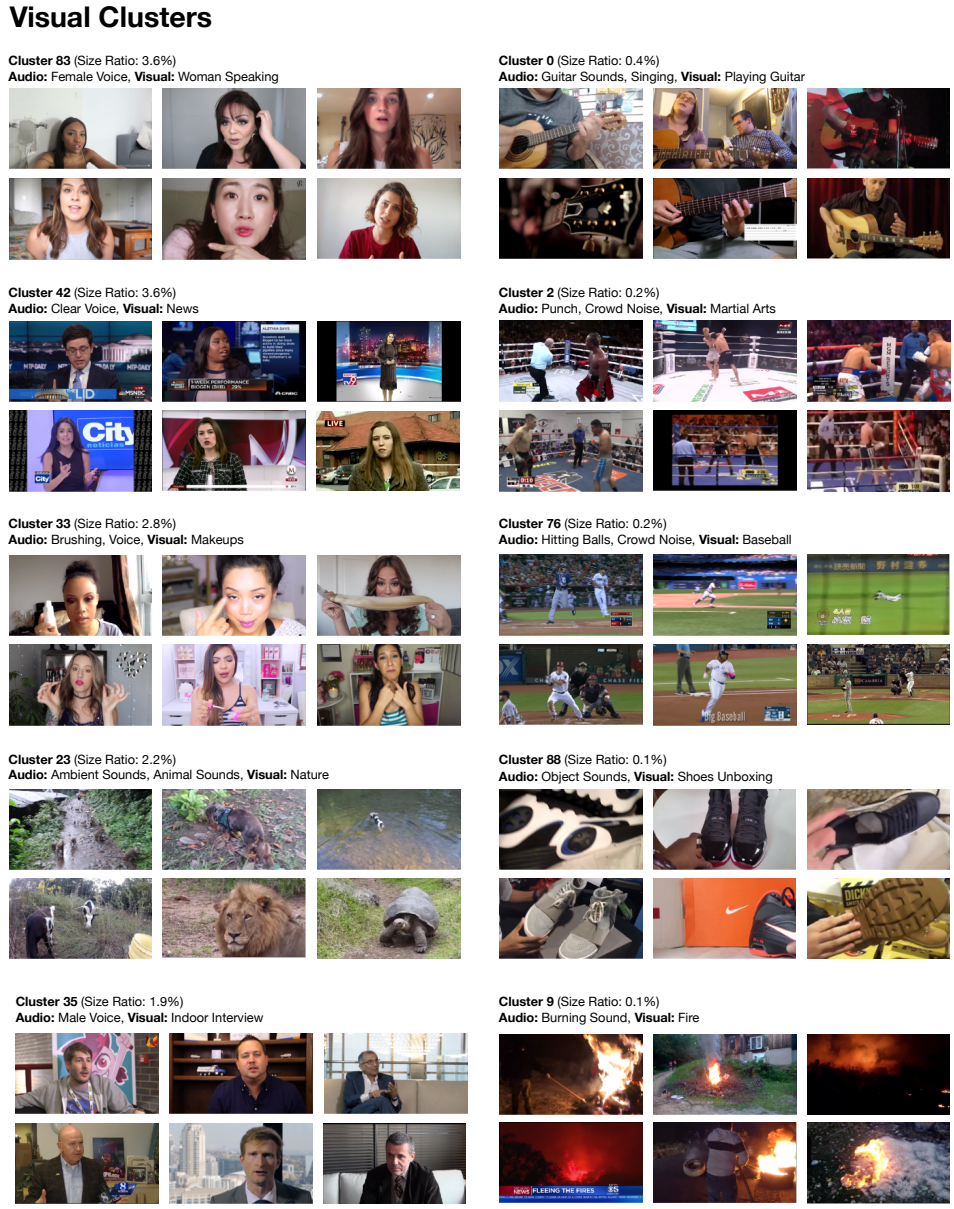


Figure 4.8: Representative samples and concepts derived from a manual inspection of 100 *visual* clusters of the 2M subset. We show samples from the five largest clusters on the left column and those from the five smallest clusters on the right. Each cluster captures distinctive audio-visual concepts, indicating that our curated subset contains various concepts with high audio-visual correspondence.

Chapter 5

Conclusion

5.1 Summary

In this thesis, we have studied several problems in self-supervised video representation learning – large compute & memory requirements and the need of scalable data curation pipeline for large-scale video datasets – and proposed novel solutions to improve learning efficiency. We summarize the main contributions of this thesis as the following.

In Chapter 2, we introduced a novel neural network for compressed video recognition, an IMR network, and trained the model on unlabeled videos without decoding them using two novel pretext tasks. Our main technical contributions include: (1) we propose the first self-supervised training approach on compressed videos, and (2) we propose a three-stream 3D CNN architecture to encode I-frames and P-Frames of compressed videos while dynamically modeling interaction between them. Our IMRNet achieved the new state-of-the-art performance on compressed video recognition in both fully-supervised and

self-supervised settings, and the proposed pretext tasks outperformed previous self-supervised approaches in downstream tasks.

In Chapter 3, we introduced a multimodal bidirectional Transformer architecture for audio-visual self-supervised video representation learning. To train our model end-to-end, we proposed a novel parameter reduction technique based on matrix decomposition with low-rank approximation, resulting in up to 97% parameter reduction. With our novel content-ware negative sampling technique, we successfully trained a multimodal Transformer end-to-end in a contrastive manner for audio-visual learning, which is the first time in the literature. Our trained model achieved competitive performance in a variety of downstream tasks.

In Chapter 4, we proposed an automatic and scalable data collection pipeline for self-supervised audio-visual representation learning. We empirically demonstrated that our MI-based subset selection algorithm can retrieve correspondence in both artificial and practical settings, and that audio and visual models trained on datasets curated with our automatic pipeline yield competitive or better performance than those trained on existing, manually verified datasets. We released a large-scale open-domain video dataset of 100M clips curated with our pipeline.

5.2 Future Work

There is still room for improvement in relation to efficient self-supervised video representation learning. In future work, we may consider improving our proposed methods in this thesis in the following ways:

Exploring More Modalities. As humans, enabling one modality to educate other modalities is key to learning to perceive a multimodal world and building a holistic understanding without an external teacher [213, 214, 215, 6].

Since this *learning from reentry* generally improves with more modalities [216], we need to explore another data source of videos in addition to visual frames and audio streams, that is, speech transcripts. Especially, raw visual frames from web videos and their corresponding (automatic) transcripts usually have weak and noisy correlations [67], so extending our data curation pipeline to the audio, vision and language setting is an interesting future work.

Reducing the Effective Length of Video Inputs. Even though we reduced the model size by 97% using our parameter reduction technique, the memory requirements of our multimodal Transformer are too large: We used a total of 64 NVIDIA V100 GPUs for training the model with a batch size of 256. These issues are applicable to not only contrastive methods used in our framework, but also other self-supervised representation learning approaches including non-contrastive Siamese methods [217, 218] and masked modeling methods [219, 220, 221]. The essence of the problem lies in that Transformers scale quadratically with the length of the input sequence, and recent vision Transformers have exacerbated the problem by taking as input a long sequence of image or video frame patches [222]. Hence, we need to explore how to effectively reduce the sequence length of video inputs and incorporate it into our training framework.

Balanced Training Recipes for Unified Multimodal Models. The multimodal models used in this thesis (Chapter 3 and Chapter 4) have modality-specific encoders. While this architecture allows us to capture the dynamics of each modality-specific input distribution, we need separate encoders whenever we add different modalities. Furthermore, we need to finetune separate prediction heads for each encoder to evaluate on downstream tasks, which is very cumbersome. Thus, we need a single unified architecture to model data of different modalities, and the success of Vision Transformer [222] opened up the possibility

towards this goal. However, we use different training recipes for each modality, ranging from data augmentations to optimization techniques [223, 224]. Thus, finding balanced training recipes that work for all modalities would be a good preliminary goal to create an optimal multimodal, unified model.

Bibliography

[1] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild," *arXiv preprint arXiv:1212.0402*, 2012.

[2] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *ACM-MM*, 2015.

[3] R. Arandjelovic and A. Zisserman, "Look, Listen and Learn," in *ICCV*, 2017.

[4] H. Chen, W. Xie, A. Vedaldi, and A. Zisserman, "VGGSound: A Large-Scale Audio-Visual Dataset," in *ICASSP*, 2020.

[5] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An Ontology and Human-Labeled Dataset for Audio Events," in *ICASSP*, 2017.

[6] L. Smith and M. Gasser, "The Development of Embodied Cognition: Six Lessons from Babies," *Artificial life*, vol. 11, no. 1-2, pp. 13–29, 2005.

[7] M. Wortsman, K. Ehsani, M. Rastegari, A. Farhadi, and R. Mottaghi, "Learning to Learn How to Learn: Self-Adaptive Visual Navigation using Meta-Learning," in *CVPR*, 2019.

[8] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, *et al.*, "RoboTHOR: An Open Simulation-to-Real Embodied AI Platform," in *CVPR*, 2020.

[9] K. Ehsani, W. Han, A. Herrasti, E. VanderBilt, L. Weihs, E. Kolve, A. Kembhavi, and R. Mottaghi, "ManipulaTHOR: A Framework for Visual Object Manipulation," in *CVPR*, 2021.

[10] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local Neural Networks," in *CVPR*, 2018.

[11] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast Networks for Video Recognition," in *ICCV*, 2019.

[12] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu, "Video Swin Transformer," in *CVPR*, 2022.

[13] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell, "Clockwork Convnets for Video Semantic Segmentation," in *ECCV*, 2016.

[14] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep Feature Flow for Video Recognition," in *CVPR*, 2017.

[15] Y. Liu, C. Shen, C. Yu, and J. Wang, "Efficient Semantic Video Segmentation with Per-frame Inference," in *ECCV*, 2020.

[16] H. Wang, W. Wang, and J. Liu, "Temporal Memory Attention for Video Semantic Segmentation," in *ICIP*, 2021.

[17] Y. Yu, J. Kim, and G. Kim, "A Joint Sequence Fusion Model for Video Question Answering and Retrieval," in *ECCV*, 2018.

[18] S. Chen, Y. Zhao, Q. Jin, and Q. Wu, "Fine-grained Video-Text Retrieval with Hierarchical Graph Reasoning," in *CVPR*, 2020.

[19] H. Luo, L. Ji, M. Zhong, Y. Chen, W. Lei, N. Duan, and T. Li, "CLIP4Clip: An Empirical Study of CLIP for End to End Video Clip Retrieval," *arXiv preprint arXiv:2104.08860*, 2021.

[20] S. Na, S. Lee, J. Kim, and G. Kim, "A Read-Write Memory Network for Movie Story Understanding," in *ICCV*, 2017.

[21] Y. Jang, Y. Song, C. D. Kim, Y. Yu, Y. Kim, and G. Kim, "Video Question Answering with Spatio-Temporal Reasoning," *International Journal of Computer Vision*, vol. 127, no. 10, pp. 1385–1412, 2019.

[22] T. M. Le, V. Le, S. Venkatesh, and T. Tran, "Hierarchical Conditional Relation Networks for Video Question Answering," in *CVPR*, 2020.

[23] A. Yang, A. Miech, J. Sivic, I. Laptev, and C. Schmid, "Just Ask: Learning to Answer Questions from Millions of Narrated Videos," in *ICCV*, 2021.

[24] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised Learning of Video Representations using LSTMs," in *ICML*, 2015.

[25] I. Misra, C. L. Zitnick, and M. Hebert, "Shuffle and Learn: Unsupervised Learning using Temporal Order Verification," in *ECCV*, 2016.

[26] Z. Luo, B. Peng, D.-A. Huang, A. Alahi, and L. Fei-Fei, "Unsupervised Learning of Long-Term Motion Dynamics for Videos," in *CVPR*, 2017.

[27] C. Gan, B. Gong, K. Liu, H. Su, and L. J. Guibas, "Geometry Guided Convolutional Neural Networks for Self-Supervised Video Representation Learning," in *CVPR*, 2018.

[28] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language Models are Few-Shot Learners," *arXiv preprint arXiv:2005.14165*, 2020.

[29] M. Patrick, Y. M. Asano, R. Fong, J. F. Henriques, G. Zweig, and A. Vedaldi, "Multi-modal Self-Supervision from Generalized Data Transformations," *arXiv preprint arXiv:2003.04298*, 2020.

[30] X. Wang and A. Gupta, "Unsupervised Learning of Visual Representations using Videos," in *ICCV*, 2015.

[31] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson, "Learning visual groups from co-occurrences in space and time," in *ICLR Workshop*, 2016.

[32] D. Jayaraman and K. Grauman, "Slow and Steady Feature Analysis: Higher Order Temporal Coherence in Video," in *CVPR*, 2016.

[33] B. Fernando, H. Bilen, E. Gavves, and S. Gould, "Self-Supervised Video Representation Learning With Odd-One-Out Networks," in *CVPR*, 2017.

[34] D. Wei, J. J. Lim, A. Zisserman, and W. T. Freeman, "Learning and Using the Arrow of Time," in *CVPR*, 2018.

[35] D. Kim, D. Cho, and I. S. Kweon, "Self-Supervised Video Representation Learning with Space-Time Cubic Puzzles," in *AAAI*, 2019.

[36] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised Representation Learning by Predicting Image Rotations," in *ICLR*, 2018.

[37] L. Jing, X. Yang, J. Liu, and Y. Tian, "Self-Supervised Spatiotemporal Feature Learning via Video Rotation Prediction," *arXiv preprint arXiv:1811.11387*, 2018.

[38] V. R. de Sa, "Learning Classification with Unlabeled Data," *NeurIPS*, 1993.

[39] A. Owens and A. A. Efros, "Audio-Visual Scene Analysis with Self-Supervised Multisensory Features," in *ECCV*, 2018.

[40] A. Piergiovanni, A. Angelova, and M. S. Ryoo, "Evolving Losses for Unsupervised Video Representation Learning," in *CVPR*, 2020.

[41] B. Korbar, D. Tran, and L. Torresani, "Cooperative Learning of Audio and Video Models from Self-Supervised Synchronization," in *NeurIPS*, 2018.

[42] P. Morgado, N. Vasconcelos, and I. Misra, "Audio-Visual Instance Discrimination with Cross-Modal Agreement," in *CVPR*, 2021.

[43] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised Feature Learning via Non-Parametric Instance Discrimination," in *CVPR*, 2018.

[44] A. Brock, J. Donahue, and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," in *ICLR*, 2019.

[45] A. Clark, J. Donahue, and K. Simonyan, "Adversarial Video Generation on Complex Datasets," *arXiv preprint arXiv:1907.06571*, 2019.

[46] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *NAACL-HLT*, 2019.

[47] C. Feichtenhofer, H. Fan, B. Xiong, R. Girshick, and K. He, "A Large-Scale Study on Unsupervised Spatiotemporal Representation Learning," in *CVPR*, 2021.

[48] E. Cole, X. Yang, K. Wilber, O. Mac Aodha, and S. Belongie, "When Does Contrastive Visual Representation Learning Work?," in *CVPR*, 2022.

[49] D. Le Gall, "MPEG: a video compression standard for multimedia applications," *Communications of the ACM*, vol. 34, no. 4, pp. 46–58, 1991.

[50] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang, "Real-time Action Recognition with Enhanced Motion Vector CNNs," in *CVPR*, 2016.

[51] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Compressed Video Action Recognition," in *CVPR*, 2018.

[52] Z. Shou, X. Lin, Y. Kalantidis, L. Sevilla-Lara, M. Rohrbach, S.-F. Chang, and Z. Yan, "DMC-Net: Generating Discriminative Motion Cues for Fast Compressed Video Action Recognition," in *CVPR*, 2019.

[53] S. Wang, H. Lu, and Z. Deng, "Fast Object Detection in Compressed Video," in *ICCV*, 2019.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *NeurIPS*, 2017.

[55] J. Howard and S. Ruder, "Universal Language Model Fine-tuning for Text Classification," in *ACL*, 2018.

[56] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep Contextualized Word Representations," in *NAACL-HLT*, 2018.

[57] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-Training," *Techincal Report, OpenAI*, 2018.

[58] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," *OpenAI Blog*, vol. 1, no. 8, 2019.

[59] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv preprint arXiv:1907.11692*, 2019.

[60] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," in *NeurIPS*, 2019.

[61] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid, "VideoBERT: A Joint Model for Video and Language Representation Learning," in *ICCV*, 2019.

[62] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei, *et al.*, "Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks," in *ECCV*, 2020.

[63] X. Hu, X. Yin, K. Lin, L. Zhang, J. Gao, L. Wang, and Z. Liu, "VIVO: Visual Vocabulary Pre-Training for Novel Object Captioning," in *AAAI*, 2021.

[64] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "ZeRO: Memory Optimizations Toward Training Trillion Parameter Models," in *SC*, 2020.

[65] G. Hinton, O. Vinyals, J. Dean, *et al.*, "Distilling the Knowledge in a Neural Network," in *NeurIPS Workshop on Deep Learning*, 2014.

[66] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *ICLR*, 2016.

[67] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic, "HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips," in *ICCV*, 2019.

[68] Y. Yu, S. Lee, G. Kim, and Y. Song, "Self-Supervised Learning of Compressed Video Representations," in *ICLR*, 2021.

[69] S. Lee, Y. Yu, G. Kim, T. Breuel, J. Kautz, and Y. Song, "Parameter Efficient Multimodal Transformers for Video Representation Learning," in *ICLR*, 2021.

[70] S. Lee, J. Chung, Y. Yu, G. Kim, T. Breuel, G. Chechik, and Y. Song, "ACAV100M: Automatic Curation of Large-Scale Datasets for Audio-Visual Video Representation Learning," in *ICCV*, 2021.

[71] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, *et al.*, "The Kinetics Human Action Video Dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[72] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A Large Video Database for Human Motion Recognition," in *ICCV*, 2011.

[73] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional Two-Stream Network Fusion for Video Action Recognition," in *CVPR*, 2016.

[74] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016.

[75] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," in *ICCV*, 2015.

[76] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," in *CVPR*, 2017.

[77] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A Closer Look at Spatiotemporal Convolutions for Action Recognition," in *CVPR*, 2018.

[78] K. Grauman and T. Darrell, "The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features," in *ICCV*, 2005.

[79] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," in *CVPR*, 2006.

[80] J. M. Prewitt *et al.*, "Object Enhancement and Extraction," *Picture processing and Psychopictorics*, vol. 10, no. 1, pp. 15–19, 1970.

[81] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," in *ICML*, 2020.

[82] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum Contrast for Unsupervised Visual Representation Learning," in *CVPR*, 2020.

[83] I. Misra and L. v. d. Maaten, "Self-Supervised Learning of Pretext-Invariant Representations," in *CVPR*, 2020.

[84] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *NeurIPS*, 2014.

[85] J. Wang, J. Jiao, L. Bao, S. He, Y. Liu, and W. Liu, "Self-supervised Spatio-temporal Representation Learning for Videos by Predicting Motion and Appearance Statistics," in *CVPR*, 2019.

[86] D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie, and Y. Zhuang, "Self-supervised Spatiotemporal Learning via Video Clip Order Prediction," in *CVPR*, 2019.

[87] T. Han, W. Xie, and A. Zisserman, "Video Representation Learning by Dense Predictive Coding," in *ICCV Workshop on Large Scale Holistic Video Understanding*, 2019.

[88] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[89] V. Kumar BG, G. Carneiro, and I. Reid, "Learning Local Image Descriptors with Deep Siamese and Triplet Convolutional Networks by Minimising Global Loss Functions," in *CVPR*, 2016.

[90] R. Santa Cruz, B. Fernando, A. Cherian, and S. Gould, "DeepPermNet: Visual Permutation Learning," in *CVPR*, 2017.

[91] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised Visual Representation Learning by Context Prediction," in *ICCV*, 2015.

[92] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy, "Tracking Emerges by Colorizing Videos," in *ECCV*, 2018.

[93] X. Wang, A. Jabri, and A. A. Efros, "Learning Correspondence from the Cycle-Consistency of Time," in *CVPR*, 2019.

[94] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation Learning with Contrastive Predictive Coding," *arXiv preprint arXiv:1807.03748*, 2018.

[95] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning Deep Representations by Mutual Information Estimation and Maximization," in *ICLR*, 2019.

[96] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in *CVPR*, 2018.

[97] M. S. Ryoo, A. Piergiovanni, J. Kangaspunta, and A. Angelova, "AssembleNet++: Assembling Modality Representations via Attention Connections," 2020.

[98] M. S. Ryoo, A. Piergiovanni, M. Tan, and A. Angelova, "AssembleNet: Searching for Multi-Stream Neural Connectivity in Video Architectures," in *ICLR*, 2020.

[99] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal Machine Learning: A Survey and Taxonomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 423–443, 2018.

[100] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba, "Learning Sight from Sound: Ambient Sound Provides Supervision for Visual Learning," *International Journal of Computer Vision*, vol. 126, no. 10, pp. 1120–1137, 2018.

[101] R. Arandjelovic and A. Zisserman, "Objects that Sound," in *ECCV*, 2018.

[102] A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. T. Freeman, and M. Rubinstein, "Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–11, 2018.

[103] R. Gao and K. Grauman, "2.5D Visual Sound," in *CVPR*, 2019.

[104] H. Alwassel, D. Mahajan, L. Torresani, B. Ghanem, and D. Tran, "Self-Supervised Learning by Cross-Modal Audio-Video Clustering," *arXiv preprint arXiv:1911.12667*, 2019.

[105] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik, "AVA: A Video Dataset of Spatio-temporally Localized Atomic Visual Actions," in *CVPR*, 2018.

[106] J. Carreira, E. Noland, C. Hillier, and A. Zisserman, "A Short Note on the Kinetics-700 Human Action Dataset," *arXiv preprint arXiv:1907.06987*, 2019.

[107] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, "Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding," in *ECCV*, 2016.

[108] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[109] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *EMNLP*, 2014.

[110] J. Lu, D. Batra, D. Parikh, and S. Lee, "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks," in *NeurIPS*, 2019.

[111] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, "VL-BERT: Pre-training of Generic Visual-Linguistic Representations," in *ICLR*, 2020.

[112] H. Tan and M. Bansal, "LXMERT: Learning Cross-Modality Encoder Representations from Transformers," in *EMNLP-IJCNLP*, 2019.

[113] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," in *NeurIPS Deep Learning Symposium*, 2016.

[114] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen, "EPIC-Fusion: Audio-Visual Temporal Binding for Egocentric Action Recognition," in *ICCV*, 2019.

[115] C. Sun, F. Baradel, K. Murphy, and C. Schmid, "Learning Video Representations using Contrastive Bidirectional Transformer," *arXiv preprint arXiv:1906.05743*, 2019.

[116] M. Gutmann and A. Hyvärinen, "Noise-Contrastive Estimation: A New Estimation Principle for Unnormalized Statistical Models," in *AISTATS*, 2010.

[117] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," in *ICLR*, 2020.

[118] D. Hendrycks and K. Gimpel, "Gaussian Error Linear Units (GELUs)," *arXiv preprint arXiv:1606.08415*, 2016.

[119] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep Image Prior," in *CVPR*, 2018.

[120] J. Nocedal and S. Wright, *Numerical Optimization*. Springer Science & Business Media, 2006.

[121] G. H. Golub and C. F. Van Loan, *Matrix Computations*, vol. 3. JHU press, 2012.

[122] B. Chen, W. L. A. Garg, Z. Yu, A. Shrivastava, J. Kautz, and A. Anandkumar, "Angular Visual Hardness," in *ICML Workshop on Identifying and Understanding Deep Learning Phenomena*, 2019.

[123] M. Lezcano-Casado and D. Martínez-Rubio, "Cheap Orthogonal Constraints in Neural Networks: A Simple Parametrization of the Orthogonal and Unitary Group," in *ICML*, 2019.

[124] S. Bai, J. Z. Kolter, and V. Koltun, "Deep Equilibrium Models," in *NeurIPS*, 2019.

[125] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *INTERSPEECH*, 2019.

[126] S. J. Reddi, S. Kale, and S. Kumar, "On the Convergence of Adam and Beyond," in *ICLR*, 2018.

[127] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *ICLR*, 2019.

[128] Y. Aytar, C. Vondrick, and A. Torralba, "SoundNet: Learning Sound Representations from Unlabeled Video," in *NeurIPS*, 2016.

[129] K. J. Piczak, "Environmental Sound Classification with Convolutional Neural Networks," in *MLSP*, 2015.

[130] D. Hu, F. Nie, and X. Li, "Deep Multimodal Clustering for Unsupervised Audiovisual Learning," in *CVPR*, 2019.

[131] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta, "Asynchronous Temporal Fields for Action Recognition," in *CVPR*, 2017.

[132] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "VisualBERT: A Simple and Performant Baseline for Vision and Language," *arXiv preprint arXiv:1908.03557*, 2019.

[133] C. Alberti, J. Ling, M. Collins, and D. Reitter, "Fusion of Detected Objects in Text for Visual Question Answering," in *EMNLP-IJCNLP*, 2019.

[134] G. Li, N. Duan, Y. Fang, M. Gong, D. Jiang, and M. Zhou, "Unicoder-VL: A Universal Encoder for Vision and Language by Cross-modal Pre-training," in *AAAI*, 2020.

[135] L. Zhou, H. Palangi, L. Zhang, H. Hu, J. J. Corso, and J. Gao, "Unified Vision-Language Pre-Training for Image Captioning and VQA," in *AAAI*, 2020.

[136] Y.-C. Chen, L. Li, L. Yu, A. E. Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu, "UNITER: UNiversal Image-TExt Representation Learning," *arXiv preprint arXiv:1909.11740*, 2019.

[137] L. Zhu and Y. Yang, "ActBERT: Learning Global-Local Video-Text Representations," in *CVPR*, 2020.

[138] H. Luo, L. Ji, B. Shi, H. Huang, N. Duan, T. Li, X. Chen, and M. Zhou, "UniViLM: A Unified Video and Language Pre-Training Model for Multimodal Understanding and Generation," *arXiv preprint arXiv:2002.06353*, 2020.

[139] W. Boes and H. Van hamme, "Audiovisual Transformer Architectures for Large-Scale Classification and Synchronization of Weakly Labeled Audio Events," in *ACM MM*, 2019.

[140] Y. Tian, D. Li, and C. Xu, "Unified Multisensory Perception: Weakly-Supervised Audio-Visual Video Parsing," in *ECCV*, 2020.

[141] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal Deep Learning," in *ICML*, 2011.

[142] N. Srivastava and R. R. Salakhutdinov, "Multimodal Learning with Deep Boltzmann Machines," in *NeurIPS*, 2012.

[143] A. Owens, P. Isola, J. McDermott, A. Torralba, E. H. Adelson, and W. T. Freeman, "Visually Indicated Sounds," in *CVPR*, 2016.

[144] H. Zhao, C. Gan, A. Rouditchenko, C. Vondrick, J. McDermott, and A. Torralba, "The Sound of Pixels," in *ECCV*, 2018.

[145] A. Miech, J.-B. Alayrac, L. Smaira, I. Laptev, J. Sivic, and A. Zisserman, "End-to-End Learning of Visual Representations from Uncurated Instructional Videos," in *CVPR*, 2020.

[146] R. Reed, "Pruning Algorithms-A Survey," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[147] M. Caron, A. Morcos, P. Bojanowski, J. Mairal, and A. Joulin, "Pruning Convolutional Neural Networks with Self-Supervision," *arXiv preprint arXiv:2001.03554*, 2020.

[148] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," in *ECCV*, 2016.

[149] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017.

[150] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, "Searching for MobileNetV3," in *ICCV*, 2019.

[151] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *CVPR*, 2018.

[152] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size," *arXiv preprint arXiv:1602.07360*, 2016.

[153] S. Bai, J. Z. Kolter, and V. Koltun, "Trellis Networks for Sequence Modeling," in *ICLR*, 2019.

[154] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and Ł. Kaiser, "Universal Transformers," in *ICLR*, 2019.

[155] R. Dabre and A. Fujita, "Recurrent Stacking of Layers for Compact Neural Machine Translation Models," in *AAAI*, 2019.

[156] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in *CVPR*, 2015.

[157] C. Zhuang, A. L. Zhai, and D. Yamins, "Local Aggregation for Unsupervised Learning of Visual Embeddings," in *ICCV*, 2019.

[158] M. Wu, C. Zhuang, M. Mosse, D. Yamins, and N. Goodman, "On Mutual Information in Contrastive Learning for Visual Representations," *arXiv preprint arXiv:2005.13149*, 2020.

[159] E. J. Gibson, *Principles of Perceptual Learning and Development*. Appleton-Century-Crofts, 1969.

[160] J. W. Fisher and T. Darrell, "Probabalistic Models and Informative Subspaces for Audiovisual Correspondence," in *ECCV*, 2002.

[161] L. Paninski, "Estimation of Entropy and Mutual Information," *Neural computation*, vol. 15, no. 6, pp. 1191–1253, 2003.

[162] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating Mutual Information," *Physical review E*, vol. 69, no. 6, 2004.

[163] M. Meilă, "Comparing Clusterings—An Information Based Distance," *Journal of multivariate analysis*, vol. 98, no. 5, 2007.

[164] N. X. Vinh, J. Epps, and J. Bailey, "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance," *Journal of Machine Learning Research*, vol. 11, 2010.

[165] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, "ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding," in *CVPR*, 2015.

[166] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8M: A Large-Scale Video Classification Benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[167] N. M. Merler, K.-N. C. Mac, D. Joshi, Q.-B. Nguyen, S. Hammer, J. Kent, J. Xiong, M. N. Do, J. R. Smith, and R. Schmidt Feris, "Automatic Curation of Sports Highlights Using Multimodal Excitement Features," *IEEE Trans Multimedia*, vol. 21, no. 5, 2019.

[168] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Science and Language*, 2019.

[169] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How Transferable are Features in Deep Neural Networks?," in *NeurIPS*, 2014.

[170] S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[171] L. Bottou and Y. Bengio, "Convergence Properties of the K-Means Algorithms," in *NeurIPS*, 1995.

[172] Y. Chen and A. Krause, "Near-optimal Batch Mode Active Learning and Adaptive Submodular Optimization," *ICML*, 2013.

[173] Y. Tian, J. Shi, B. Li, Z. Duan, and C. Xu, "Audio-Visual Event Localization in Unconstrained Videos," in *ECCV*, 2018.

[174] J. Roth, S. Chaudhuri, O. Klejch, R. Marvin, A. Gallagher, L. Kaver, S. Ramaswamy, A. Stopczynski, C. Schmid, Z. Xi, and C. Pantofaru, "AVA-ActiveSpeaker: An Audio-Visual Dataset for Active Speaker Detection," *arXiv preprint arXiv:1901.01342*, 2019.

[175] S. Chaudhuri, J. Roth, D. P. W. Ellis, A. Gallagher, L. Kaver, R. Marvin, C. Pantofaru, N. Reale, L. Guarino Reid, K. Wilson, and Z. Xi, "AVA-Speech: A Densely Labeled Dataset of Speech Activity in Movies," in *Interspeech*, 2018.

[176] K. Yang, B. Russell, and J. Salamon, "Telling Left From Right: Learning Spatial Correspondence of Sight and Sound," in *CVPR*, 2020.

[177] M. Monfort, A. Andonian, B. Zhou, K. Ramakrishnan, S. A. Bargal, T. Yan, L. Brown, Q. Fan, D. Gutfruend, and C. Vondrick, "Moments in Time Dataset: One Million Videos for Event Understanding," *PAMI*, pp. 1–8, 2019.

[178] J. C. Stroud, D. A. Ross, C. Sun, J. Deng, R. Sukthankar, and C. Schmid, "Learning Video Representations from Textual Web Supervision," *arXiv preprint arXiv:2007.14937*, 2020.

[179] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core Vector Machines: Fast SVM Training on Very Large Data Sets," *Journal of Machine Learning Research*, vol. 6, no. Apr, pp. 363–392, 2005.

[180] K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes, "Using Document Summarization Techniques for Speech Data Subset Selection," in *NAACL*, 2013.

[181] Y. Shinohara, "A Submodular Optimization Approach to Sentence Set Selection," in *ICASSP*, 2014.

[182] K. Wei, Y. Liu, K. Kirchhoff, C. Bartels, and J. Bilmes, "Submodular Subset Selection for Large-Scale Speech Training Data," in *ICASSP*, 2014.

[183] S. Har-Peled and S. Mazumdar, "On Coresets for K-Means and K-Median Clustering," in *STOC*, 2004.

[184] K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes, "Unsupervised Submodular Subset Selection for Speech Data," in *ICASSP*, 2014.

[185] D. D. Lewis and W. A. Gale, "A Sequential Algorithm for Training Text Classifiers," in *SIGIR*, 1994.

[186] B. Settles, "Active Learning Literature Survey," *Science*, vol. 10, no. 3, pp. 237–304, 1995.

[187] K. Wei, R. Iyer, and J. Bilmes, "Submodularity in Data Subset Selection and Active Learning," in *ICML*, 2015.

[188] M. Minoux, "Accelerated Greedy Algorithms for Maximizing Submodular Set Functions," in *Optimization techniques*, pp. 234–243, Springer, 1978.

[189] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An Analysis of Approximations for Maximizing Submodular Set Functions–I," *Mathematical Programming*, vol. 14, no. 1, p. 265–294, 1978.

[190] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan, "Geometric Approximation via Coresets," *Combinatorial and Computational Geometry*, vol. 52, pp. 1–30, 2005.

[191] Y. Guo, "Active Instance Sampling via Matrix Partition," in *NeurIPS*, 2010.

[192] X. Li and Y. Guo, "Adaptive Active Learning for Image Classification," in *CVPR*, 2013.

[193] J. Sourati, M. Akcakaya, J. G. Dy, T. K. Leen, and D. Erdogmus, "Classification Active Learning Based on Mutual Information," *Entropy*, vol. 18, no. 2, p. 51, 2016.

[194] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "FastText.zip: Compressing Text Classification Models," *arXiv preprint arXiv:1612.03651*, 2016.

[195] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of Tricks for Efficient Text Classification," in *EACL*, 2017.

[196] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis, "How Easy Is Local Search?," *Journal of computer and system sciences*, vol. 37, no. 1, pp. 79–100, 1988.

[197] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, "CNN Architectures for Large-Scale Audio Classification," in *ICASSP*, 2017.

[198] J. Walters-Williams and Y. Li, "Estimation of Mutual Information: A Survey," in *RSKT*, 2009.

[199] M. Wang and W. Deng, "Deep Visual Domain Adaptation: A Survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[200] D. Bau, B. Zhou, A. Oliva, and A. Torralba, "Interpreting Deep Visual Representations via Network Dissection," *PAMI*, vol. 41, no. 9, pp. 2131–2145, 2019.

[201] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An Analysis of Approximations for Maximizing Submodular Set Functions—I," *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

[202] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, *et al.*, "Top 10 Algorithms in Data

Mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[203] T. Martinetz and K. Schulten, "A "Neural-Gas" Network Learns Topologies," in *ICANN*, 1991.

[204] D. Sculley, "Web-Scale K-Means Clustering," in *WWW*, 2010.

[205] A. Krizhevsky and G. Hinton, "Learning Multiple Layers of Features from Tiny Images," tech. rep., University of Toronto, 2009.

[206] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[207] Z. Jackson, C. Souza, J. Flaks, Y. Pan, H. Nicolas, and A. Thite, "Free Spoken Digit Dataset: v1.0.8," Aug. 2018.

[208] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR*, 2009.

[209] K. Pearson, "LIII. On Lines and Planes of Closest Fit to Systems of Points in Space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[210] C. Feichtenhofer, A. Pinz, and R. Wildes, "Spatiotemporal Residual Networks for Video Action Recognition," in *NeurIPS*, 2016.

[211] J. L. Fleiss, "Measuring Nominal Scale Agreement Among Many Raters," *Psychological Bulletin*, vol. 76, no. 5, p. 378, 1971.

[212] J. R. Landis and G. G. Koch, "The Measurement of Observer Agreement for Categorical Data," *Biometrics*, pp. 159–174, 1977.

[213] J. Piaget and M. T. Cook, *The Origins of Intelligence in Children*. WW Norton & Co, 1952.

[214] G. M. Edelman, "Neural Darwinism: Selection and Reentrant Signaling in Higher Brain Function," *Neuron*, vol. 10, no. 2, pp. 115–125, 1993.

[215] R. S. Chapman, "Children's Language Learning: An Interactionist Perspective," *The Journal of Child Psychology and Psychiatry and Allied Disciplines*, vol. 41, no. 1, pp. 33–54, 2000.

[216] R. Zellers, J. Lu, X. Lu, Y. Yu, Y. Zhao, M. Salehi, A. Kusupati, J. Hessel, A. Farhadi, and Y. Choi, "MERLOT Reserve: Multimodal Neural Script Knowledge through Vision and Language and Sound," in *CVPR*, 2022.

[217] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, *et al.*, "Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning," in *NeurIPS*, 2020.

[218] X. Chen and K. He, "Exploring Simple Siamese Representation Learning," in *CVPR*, 2021.

[219] H. Bao, L. Dong, and F. Wei, "BEiT: BERT Pre-Training of Image Transformers," in *ICLR*, 2022.

[220] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked Autoencoders Are Scalable Vision Learners," in *CVPR*, 2022.

[221] Z. Tong, Y. Song, J. Wang, and L. Wang, "VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training," in *NeurIPS*, 2022.

[222] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *ICLR*, 2021.

[223] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, *et al.*, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[224] X. Chen, S. Xie, and K. He, "An Empirical Study of Training Self-Supervised Vision Transformers," in *ICCV*, 2021.

요약

비디오는 학습에 사용할 수 있는 동적이고 멀티모달(multimodal)한 시그널을 제공하기 때문에 컴퓨터비전과 기계학습에 있어서 아주 매력적인 데이터원이다. 특히 비디오 라벨링에는 시간적, 금전적 비용이 많이 들기 때문에 최근에는 비디오 이해를 위해 자기지도 비디오 표현학습법이 많이 주목받고 있다. 하지만 자기지도학습법은 주로 대규모 학습으로 진행되어 많은 연산 및 메모리 자원이 필요로 한다. 또한, 우리가 구할 수 있는 현실 속 비디오들은 대부분 노이즈를 많이 담고 있어 인간이 별도로 검수하지 않는 한 학습에 사용하기 좋은 비디오 데이터를 구하기가 어려워 대규모 데이터 수집에 어려움이 있다.

본 학위논문에서는, 위에서 언급된 자기지도 비디오 표현학습법과 관련된 문제들을 심층적으로 알아보고 학습의 효율성을 증대시키기 위한 세 가지 해결책을 제시한다. 첫 번째로, 라벨이 달리지 않은 비디오를 별도의 디코딩 과정 없이 학습에 사용할 수 있는 방법을 알아본다. 비디오는 보통 MPEG와 같은 압축된 형식으로 저장이 되고, 이를 디코딩하기 위해서는 많은 연산 자원이 필요하다. 본 논문에서 제시하는 새로운 모델 구조와 pretext task들은 최소한의 성능 감소만으로 압축된 형태의 비디오에서 학습이 가능하게 해주며 디코딩을 생략하여 빠른 비디오 처리를 가능하게 해준다. 두 번째로, 라벨이 달리지 않은 비디오로부터 문맥화된 청각-시각 표현을 자기지도학습으로 배우기 위한 양방향 멀티모달 트랜스포머(Transformer) 구조를 제시한다. 트랜스포머 모델이 많은 메모리 자원을 소요하기 때문에 기존에 멀티모달 트랜스포머 모델은 대규모로 종단간 학습(end-to-end training)을 진행하기가 어려웠다. 저차원 근사법에 기반한 행렬 분해를 통해 본 논문에서는 멀티모달 트랜스포머 모델의 크기를 줄여 성공적으로 종단간 학습시켰으며, 다양한 태스크에서 좋은 성능을 거두었다. 마지막으로, 청각-시각 자기지도 표현학습법에 사용할 수 있는 비디오 데이터를 모으기 위한 확장 가능하

고(scalable) 자동화된 수집 파이프라인을 제안한다. 이 파이프라인은 상호정보량(Mutual Information)에 기반한 부분 집합 선택 알고리즘을 통해 노이즈가 있는 데이터를 필터링하며, 이를 통해 수집된 데이터셋에서 학습된 청각 및 시각 모델들은 인간의 검수를 통해 만들어진 기존 데이터셋에서 학습된 모델과 비교하여 비슷하거나 더 나은 성능을 보인다. 본 논문에서는 이 파이프라인을 이용하여 청각 및 시각 표현학습을 위해 사용할 수 있는, 1억 개의 비디오 클립으로 구성된 오픈 도메인 비디오 데이터셋 ACAV100M을 구성하였다.

주요어: 딥러닝, 컴퓨터비전, 대규모 비디오 이해, 자기지도 표현학습법
학번: 2017-26247

Acknowledgements

I would like to express my sincere gratitude to my advisor Prof. Gunhee Kim who has given constant encouragement and valuable advice over the past six years. I would also like to thank my mentor, Yale Song, for teaching me the attitude that I should have as a researcher. It will be a lifelong help for me to live as a researcher in the future. I also greatly appreciate my thesis committee – Prof. Sungjoo Yoo, Prof. U Kang, Prof. Joonseok Lee and Prof. Sung-Ju Hwang – for their valuable guidance on writing a good academic thesis.

None of my work would have been possible without my collaborators: Seil Na, Youngjae Yu, Jisung Kim, Joonil Na, Jaeyoun Kang, Jinyoung Sung, Thomas Breuel, Jan Kautz, Jiwan Chung and Gal Chechik. I would also like to mention all of the labmates in SNU Vision and Learning Laboratory: Junhyug Noh, Yunseok Jang, Jongwook Choi, Juyong Kim, Hanseul Jun, Yookoon Park, Byeongchang Kim, Youngjin Kim, Cesc Chunseong Park, Soochan Lee, Wonhee Lee, Insu Jeon, Minui Hong, Hyouk Jang, Jongseok Kim, Minjung Kim, Jaemin Cho, Amelie Schmidt-Colberg, Taeyoung Hahn, Wonkwang Lee, Myeongjang Pyeon, Chris Dongjoo Kim, Hyunwoo Kim, Jaekyeom Kim, Jaewoo Ahn, Junsoo Ha, Sungeun Kim, Jinseo Jeong, Heeseung Yun, Sangwoo Moon, Dongyeon Woo, Seungyoon Woo, Jihwan Moon, Seokhee Hong, Eunkyu Park, Dohyun

Kim, Junseo Koo, Keighley Overbay, Sehun Lee, Yeda Song, Hyungyu Seo, Dayoon Ko, Fatemeh Pesaran and Julian Paquerot. I will never forget the precious time that we spent together.

Last but not least, I would like to thank my family for their love, support, and encouragement.