Reducing Annotation and Computation Costs for Efficient Compressed Video Action Recognition

Hayato Terao
Graduate School of Information Science and Technology,
Hokkaido University, Japan

January, 2024

Acknowledgement

I want to express my gratitude to my advisers, Prof. Masahito Yamamoto, Specially Appointed Associate Prof. Hiroyuki Iizuka, and Specially Appointed Assistant Prof. Wataru Noguchi, for their encouragement and instruction in completing this thesis.

I am also profoundly grateful to the committee of this thesis: Prof. Itsuki Noda, Specially Appointed Prof. Tetsuo Ono, and Prof. Hidenori Kawamura for their insightful critiques and suggestions. Based on their comments, the description and organization of this thesis were refined to show the contribution of this work clearly.

I also thank all the members of my research group, the Autonomous Systems Engineering Laboratory. Conversations with them always encouraged me to continue this research.

Finally, I am grateful to my family for their emotional and financial support, which enabled me to advance to the doctoral program at Hokkaido University.

This work was supported by JST SPRING Grant Number JPMJSP2119.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Video understanding is a research field that leverages machine learning to analyze video data. This technology has many applications, including automated driving, surveillance systems, and video generation. In recent years, the number of videos available online has increased due to the increase in social media and video-sharing service users worldwide. The uploaded uncountable videos have rapidly progressed video understanding research in the last decade.

A typical task in video understanding is video classification (a.k.a. action recognition), which involves classifying short video clips. In traditional studies, handcrafted descriptors like dense trajectories [Wang et al., 2013] and improved dense trajectories [Wang and Schmid, 2013] were developed to classify videos. Recent studies have used deep networks to classify videos directly from RGB frame sequences [Hara et al., 2017, Tran et al., 2018, Feichtenhofer et al., 2019].

Accessing RGB frames requires decoding because most videos are compressed for efficient storage and transmission. While some studies have even incorporated optical flow as an auxiliary input to boost accuracy [Wang et al., 2018,

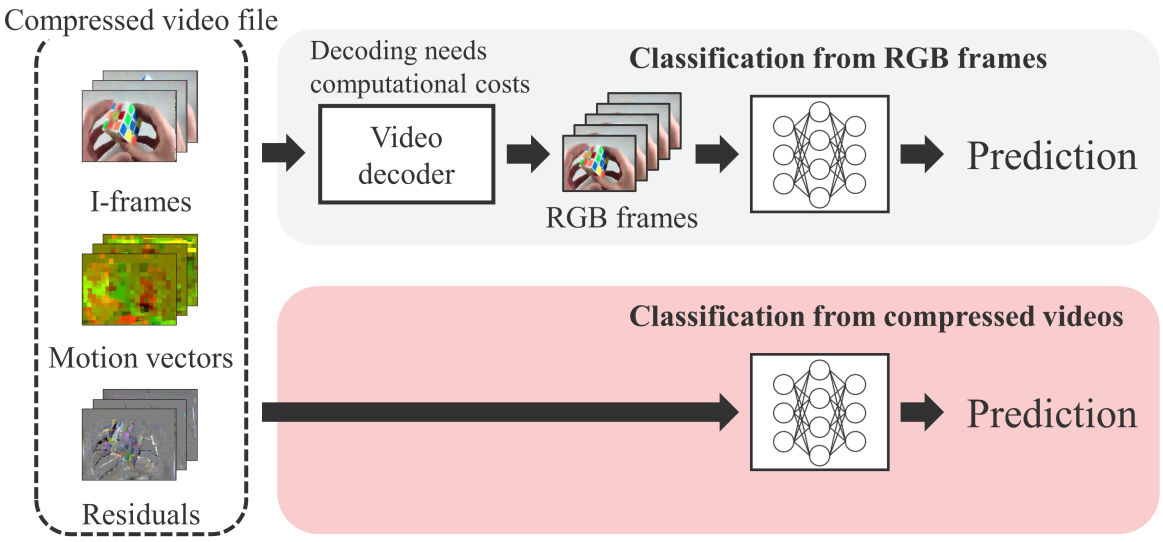


Fig. 1.1 Comparison with popular RGB-based video classification and compressed video action recognition.

Simonyan and Zisserman, 2014, Lin et al., 2019, Carreira and Zisserman, 2017, Ryoo et al., 2019, Ryoo et al., 2020], it also has the same limitation because it is computed from RGB frame sequences. This decoding limits the deployment of action recognition models on mobile or edge devices.

Videos frequently contain redundant information, such as backgrounds, and video compression reduces such redundancy by converting RGB frames into different features, including I-frames, motion vectors, and residuals. The I-frames are stored as images, whereas the motion vectors and residuals represent only the changes from previous RGB frames. Compressed video action recognition directly classifies actions from compressed video features, as depicted in Fig. 1.1. This approach has two advantages. First, we can omit the decoding process and reduce the computational cost by using compressed video features as inputs. Second, compressed video features are expected to be easier to classify than raw videos because video compression reduces the redundancy of raw videos.

Wu et al. [Wu et al., 2018] showed that this method achieves competitive classi-

fication accuracy against conventional RGB-based action recognition but with lower computational complexity defined by floating-point operations (FLOPs). Some studies have leveraged the low computational complexity of compressed action recognition to deploy action recognition models on mobile and edge devices [He et al., 2021, Huo et al., 2019].

1.2 Research Objective

Our research objective in this thesis is to improve further the efficiency of compressed video action recognition for low-resource environments. Although compressed video action recognition is more efficient than RGB frame-based video classification and suitable for mobile and edge devices, it still struggles with expensive costs to optimize and deploy deep networks. We aim to reduce such costs of compressed video action recognition, making it easier to use for many applications. Below, we describe the costs of compressed video action recognition in detail.

Annotation cost

The previous compressed video action recognition models have been optimized in a supervised manner. This approach requires large-scale annotated datasets to avoid over-fitting of deep networks. This is a critical problem because annotating many videos manually is expensive and tedious, and preparing such datasets is challenging.

Fortunately, the annotation cost problem is general in deep learning, and semi-supervised learning (SSL) is widely studied to overcome the problem. SSL optimizes deep networks using both labeled and unlabeled data. Effective SSL methods can fully utilize unlabeled data to learn helpful features for classification, reducing the number of labeled data to achieve practical accuracy. Pseudo-labeling is the widely used approach of SSL. Given unlabeled data, pseudo-labeling feeds the given data into the deep networks and generates pseudo-labels of the unlabeled data from the

predictions. The assigned pseudo labels are used as alternatives to the actual labels, and deep networks are optimized using unlabeled data and pseudo labels.

This thesis addressed the annotation cost problem by extending the pseudo-labeling method into compressed video action recognition. We first applied the simple pseudo-labeling method proposed by Lee [Lee et al., 2013] to compressed video action recognition. We found that combining Lee's method and compressed video action recognition achieved better accuracy than combining various SSL methods and RGB-based video classification. From this observation, we proposed compressed video ensembling-based pseudo labeling (CoVEnPL), a novel SSL method for compressed video action recognition. The main idea of CoVEnPL is that generating accurate pseudo-labels leads to obtaining robust models. To obtain such pseudo-labels, CoVEnPL optimized three deep networks corresponding to I-frames, motion vectors, and residuals and ensemble their predictions for pseudo-labels generation. CoVEnPL combined this ensemble approach with Fixmatch [Sohn et al., 2020], a state-of-the-art SSL method for image classification. Our experiments showed that CoVEnPL achieved better accuracy than most semi-supervised video classification methods. In particular, our method showed much better than previous studies that only used RGB frames as inputs with a shorter inference time.

Computational cost

The second focus of this thesis is the computational cost problem. In compressed video action recognition, researchers have proposed various models that reduce the computational costs of compressed video action recognition. These studies mainly focused on using lightweight networks as the backbones of three networks corresponding to I-frames, motion vectors, and residuals.

The computational complexity of multiple networks may be unnecessary because most parameters in deep networks are unused and removable after training, as shown in various studies [Han et al., 2015, Jin et al., 2016, Han et al., 2016]. For the effi-

cient ensemble of image classification, the multi-input multi-output (MIMO) model utilized the unused parameters by creating independent subnetworks within a single network [Havasi et al., 2021]. The subnetworks process multiple images independently with one feedforwarding step of the parent network and make different predictions from the images. As a result, the MIMO model achieved competitive accuracy against multiple networks while reducing the computational complexity.

Inspired by the MIMO model, we proposed a multi-stream single network (MussNet) for efficient compressed video action recognition. This model trains a single network in the MIMO manner and creates three independent subnetworks within a single network. The subnetworks in the MussNet model independently process I-frames, motion vectors, and residuals instead of the multiple networks used in the previous methods. While we can also naively optimize the single network-based models, such naive optimization results in performance degeneration. On the other hand, our proposed MussNet model did not suffer from such performance degeneration and achieved competitive accuracy against multiple networks while reducing the overall computational complexity.

1.3 Outline of the Thesis

The rest of this thesis is organized as follows. Chapter 2 introduces related studies on video classification and compressed video action recognition.

Chapter 3 addresses the annotation cost problem of compressed video action recognition by our SSL methods. In this chapter, we first describe the SSL methods for image and video classification as the background. Then, we propose SSL methods for compressed video action recognition and show their effectiveness.

Chapter 4 addresses the computational cost problem by introducing the efficient ensemble method to compressed video action recognition. This chapter first describes the ensemble methods as the background and then describes our proposed MussNet

model. We compare the MussNet model with our baseline methods and previous efficient compressed video action recognition methods to evaluate our model.

Chapter 5 summarizes this thesis and discusses what the experimental results can explain. We also describe the future perspective of this study on efficient compressed video action recognition.

Chapter 2

Video Classification and Compressed Videos

2.1 Classification Problem

In machine learning, the classification problem refers to assigning a label or category to given data based on features. This problem aims to learn the mapping of input data to predefined categories or classes.

More specifically, a classification problem involves training a machine learning model to predict the class or category of new, unseen data based on the patterns and relationships learned from a training dataset. Classification problems are common in many applications, such as image recognition, speech recognition, natural language processing, etc. Researchers have developed algorithms and models that can accurately classify data and generalize well to new, unseen data.

Formally, we have a set of classes $\{c_i | 1 \leq i \leq C\}$, where c_i is the i-th class and C is the number of classes. To consider each class c_i as the probability distribution, c_i

is converted to the one-hot encoded vector $t_i \in \mathbb{R}^C$ defined as:

$$t_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

where t_{ij} is the j-th element of t_i. Given an input x and its target class c_i, the classification model with parameters θ outputs the probability distributions $p_\theta(\hat{c}|x)$ which satisfy the following conditions:

$$p_\theta(\hat{c}|x) \geq 0, \tag{2.2}$$

$$\sum_{i=1}^{C} p_\theta(\hat{c}|x) = 1. \tag{2.3}$$

In many cases, to obtain such predictions $p_\theta(\hat{c}|x)$, the classification model outputs C scalars $\{y_1, y_2, ..., y_C\}$ and normalizes the scalars using the softmax function as follows:

$$p_\theta(\hat{c}_i|x) = \frac{\exp y_i}{\sum_{k=1}^{C} \exp y_k}. \tag{2.4}$$

Then, the network parameters θ are optimized to minimize the cross-entropy loss between $p_\theta(\hat{t}_i|x)$ and t_i, defined as follows:

$$H(p_\theta(\hat{c}|x), t_i) := -\sum_{j=1} t_{ij} \log p_\theta(\hat{c_j}|x). \tag{2.5}$$

2.2 Video Classification

Video classification is the classification problem of video data. Video data consists of consecutive RGB frames, and capturing spatiotemporal features is critical for classification. Traditional methods have classified videos from hand-crafted features, such as spatiotemporal interest points [Laptev and Lindeberg, 2003], dense trajectories [Wang et al., 2013], improved dense trajectories (iDT) [Wang and Schmid, 2013], SIFT-3D [Scovanner et al., 2007], HOG3D [Klaser et al., 2008], Motion Boundary Histogram [Dalal et al., 2006], Action Bank [Sadanand and Corso, 2012],

Cuboids [Dollár et al., 2005], 3D SURF [Willems et al., 2008], and Dynamic-Poselets [Wang et al., 2014]. These features represent different spatiotemporal information, and iDT is widely considered one of the best features for classification.

Deep networks have shown remarkable progress in computer vision for the last decade, and the trend of video classification has also shifted from traditional hand-crafted approaches to deep learning approaches. Unlike the traditional hand-crafted approaches, most deep networks receive consecutive RGB frames as inputs and learn how to capture spatiotemporal features necessary to classify the input videos. To capture practical spatiotemporal features for classification, researchers have developed various deep network architectures, such as 2D convolutional neural networks (ConvNets) and 3D ConvNets, and improved the accuracy of video classification. However, such models are over-parameterized and suffer from over-fitting. Therefore, large-scale annotated datasets have also been crucial for video classification with deep networks to avoid over-fitting. This section introduces deep network architectures and public large-scale datasets for video classification.

2.2.1 Deep Network Models

Since AlexNet [Krizhevsky et al., 2012] was developed for images, ConvNets have been widely used in the computer vision fields. Video classification models have also employed ConvNets because the video data is a sequence of images. However, unlike image classification models, video classification models should capture both spatial and temporal features within videos. Therefore, designing the temporal modeling is a critical problem in developing video classification models. There are two popular architectures: 2D ConvNets and 3D ConvNets. We summarize these architectures below.

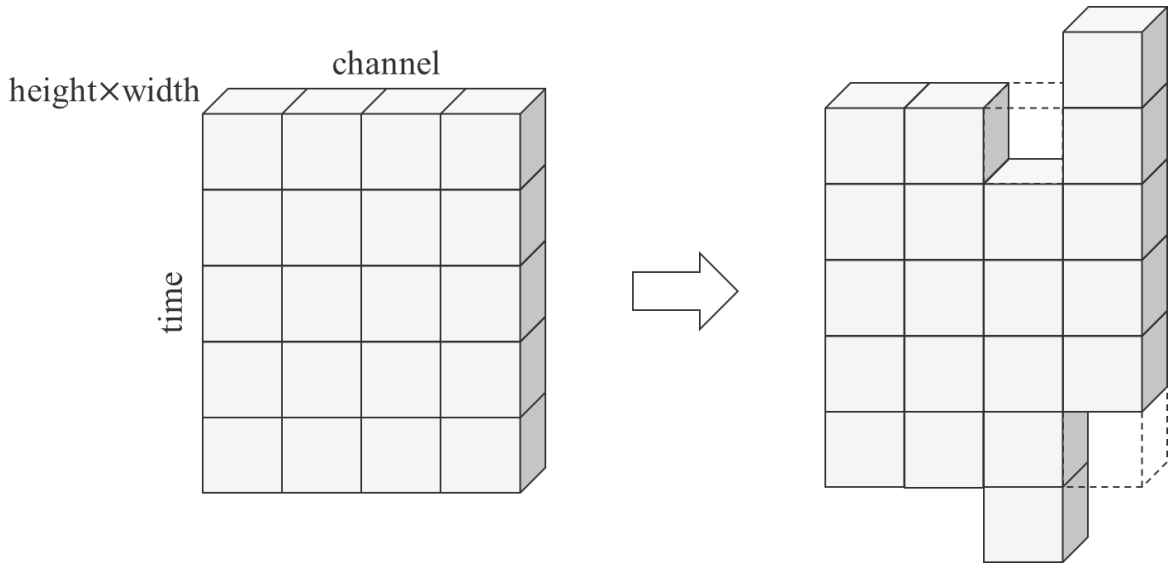


Fig. 2.1 The shift operation used in the temporal shift modules.

2D ConvNets

Before very large-scale video datasets such as Kinetics appeared, 2D ConvNets were widely used for video classification. One advantage of 2D ConvNet-based models is that their parameters can be pretrained by large-scale image datasets (e.g., ImageNet) so that the models can be fine-tuned on relatively small video datasets. Some early studies concatenated multiple frames along their channel dimensions for temporal modeling and fed the stacked frames into 2D ConvNets. [Karpathy et al., 2014, Simonyan and Zisserman, 2014, Feichtenhofer et al., 2016a] Others classified videos from each frame using 2D ConvNets and aggregated the predictions to obtain the final prediction [Wang et al., 2018]. However, the temporal modeling of these methods was not powerful, and their classification performances were limited. Thus, recent studies have employed recurrent neural networks (RNNs) to achieve more helpful temporal modeling for video classification [Sharma et al., 2015, Ballas et al., 2015, Yue-Hei Ng et al., 2015].

Today, 2D ConvNets are not the mainstream architectures for video classification,

compared with 3D ConvNets. However, some studies still employ 2D ConvNets as their backbone networks because 2D ConvNets tend to be more efficient than 3D ConvNets and Transformers regarding parameter size, computation time, and computational complexity. For example, the temporal shift module (TSM) [Lin et al., 2019], a particular case of RNNs, processes each frame by 2D ConvNets but shifts parts of channels to the neighbor frames at some points. As depicted in Fig 2.1, the shift operation can process multiple frames simultaneously with zero FLOPs, leading to powerful yet efficient time modeling. Liu et al. proposed the temporal adaptive module (TAM) [Liu et al., 2021b], an extended version of TSMs. While TSM constantly shifts channels by the pre-defined number of frames, their module dynamically determines how many frames channels should be shifted by, depending on the input videos. RubiksNet [Fan et al., 2020] also shifts channels to not only the temporal dimension but also the height and width dimensions.

3D ConvNets

Another popular approach for ConvNet-based video classification is 3D ConvNets, which use 3D convolutional kernels. Unlike 2D ConvNets, 3D ConvNets directly capture spatiotemporal features using 3D kernels. The early 3D ConvNets suffered from over-fitting because 3D kernels have more parameters than 2D convolutional kernels, and there were only relatively small datasets for video classification. For example, convolutional 3D (C3D) consists of only five layers to reduce over-fitting. Pseudo-3D residual network (P3D) reduces the parameters by approximating 3D convolutional kernels using 1D and 2D convolutional kernels.

The practical 3D ConvNets in terms of accuracy have appeared since very large-scale video datasets such as Kinetics [Carreira and Zisserman, 2017] were publicly published. The large-scale datasets provided enough videos to optimize 3D ConvNets without over-fitting, allowing researchers to develop deep 3D ConvNets. Such 3D ConvNets first employed architectures of image classification models where their

2D convolutional kernels are replaced with 3D convolutional kernels. For example, 3D ResNet (R3D) [Hara et al., 2017] is developed based on ResNet [He et al., 2016] and Inflated 3D ConvNet (I3D) [Carreira and Zisserman, 2017] is modified from GoogLeNet [Szegedy et al., 2015], a 2D ConvNet using Inception modules. Tran et al. revisited the combination of 1D and 2D convolutional kernels as alternatives to 3D convolutional kernels and proposed R(2+1)D [Tran et al., 2018]. Unlike P3D, R(2+1)D keeps the number of parameters as 3D convolutional kernels and improves accuracy from the 3D ConvNets. More recently, some 3D ConvNet architectures such as X3D [Feichtenhofer, 2020], S3D-G [Xie et al., 2018], Tiny Video Networks [Piergiovanni et al., 2022], and MoViNets [Kondratyuk et al., 2021] have been found by manual way or network architecture search (NAS) for more efficient yet robust video classification.

Multi-stream model

It is well-known that some auxiliary inputs, in addition to RGB frames, help boost video classification accuracy. Dual TV-L1 optical flow, which represents the movements of objects between successive RGB frames, is one of the most popular auxiliary inputs. Two-stream ConvNet [Simonyan and Zisserman, 2014] is an early video classification model that introduces optical flow as auxiliary inputs. This model employs late fusion, using two 2D ConvNets that classify videos from RGB frames and optical flow, respectively. Late fusion first optimizes these ConvNets independently and averages their predictions to obtain the final prediction. The two-stream I3D proposed by Carreira and Zisserman also employed late fusion but used I3D as backbone networks instead of 2D ConvNets. While late fusion has been widely used for multi-stream models and has shown accuracy improvement for various network architectures [Wang et al., 2018, Lin et al., 2019, Tran et al., 2018], some studies explored more effective fusion approaches. Feichtechofer et al. showed that intermediate fusion, which fuses optical flow features at the intermediate computa-

tion of the ConvNet of RGB frames, is better than late fusion in terms of accuracy [Feichtenhofer et al., 2016a, Feichtenhofer et al., 2016b].

The problem with optical flow is that it takes a long time to compute it from consecutive frames, even when GPU is used. To avoid this high computation, researchers have explored removing optical flow. For example, motion-augmented RGB stream (MARS) [Crasto et al., 2019] first trains ConvNet, classifying videos from optical flow. Then, MARS optimizes another ConvNet that receives RGB frames and computes similar embeddings to the optical flow ConvNet. ActionFlowNet [Ng et al., 2018] and distilled 3D networks (D3D) [Stroud et al., 2020] have two prediction heads, where one head classifies videos, and another predicts optical flow. These models use optical flow as teaching signals, making it unnecessary during the inference phase. The hidden two-stream ConvNets [Zhu et al., 2017] does not need optical flow anymore. This model employs MotionNet, which estimates optical flow from RGB frames. The MotionNet can be optimized in an unsupervised manner, i.e., pre-computed optical flow is not used for training. Then, the estimated optical flow is used as an alternative input to the actual optical flow in the multi-stream model.

The SlowFast network [Feichtenhofer et al., 2019] is also a multi-stream model that does not use optical flow. Instead of RGB frame and optical flow streams, the SlowFast network uses low- and high-FPS RGB frames. The authors claimed that low-FPS RGB frames can be considered spatial features, and high-FPS RGB frames can be considered temporal features. Under the assumptions, low-FPS RGB frames are processed by the high-capacity network with a narrow temporal receptive field. In contrast, high-FPS RGB frames are processed by the low-capacity network with a wide temporal receptive field. The SlowFast network also employed intermediate fusion, achieving state-of-the-art video classification accuracy.

Fig. 2.2 Video examples of the UCF-101 dataset.

2.2.2 Public Datasets

Deep networks have many parameters and tend to suffer from over-fitting. To avoid over-fitting, large-scale annotated datasets are required. We use the following video datasets in our experiments to evaluate our methods.

UCF-101

The UCF-101 dataset [Soomro et al., 2012] (Fig. 2.2) consists of 13,320 video clips across 101 action categories. All clips are collected from YouTube and have a fixed 25 FPS with a resolution of 320×240. This dataset provides train-test splits for three-fold cross-validation, where each split uses about 9,600 videos for training and others for validation. Note that each split specifies a slightly different number of videos for training and validation, but the number of clips per action category is almost balanced.

HMDB-51

The HMDB-51 dataset [Kuehne et al., 2011] (Fig. 2.3) consists of 6,770 video clips across 51 action categories, where each action category has a minimum of 101 clips

Fig. 2.3 Video examples of the HMDB-51 dataset.

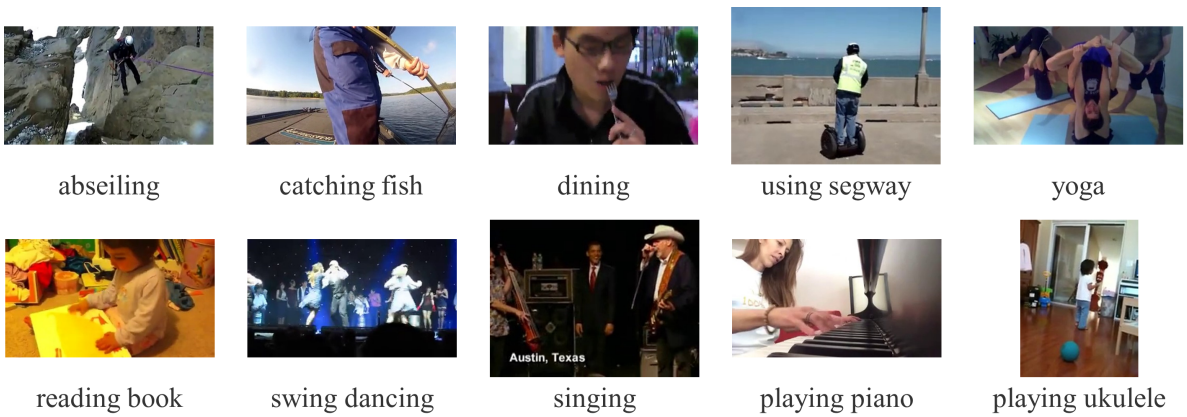


Fig. 2.4 Video examples of the Kinetics dataset.

collected from various sources, including movies and public databases. Although video clips have various FPS, we converted them to 25 FPS. This dataset also provides train-test splits for three-fold cross-validation. Each split specifies 70 clips for training and 30 clips for validation per category. Therefore, the number of clips per action category in train-test splits is perfectly balanced with a ratio of 70:30.

Kinetics

The Kinetics dataset [Carreira and Zisserman, 2017] (Fig. 2.4) is first proposed as a set of 306,245 video clips from 400 action categories, where each category has a mini-

mum of 400 clips taken from different YouTube videos. Each clip lasts around 10 seconds and has various FPS and resolutions. The actions are human-focused and cover a broad range of classes, including human-object interactions such as playing instruments and human-human interactions such as shaking hands. This dataset is regularly extended, and some variants called Kinetics-600 [Carreira et al., 2018], which consists of 480K videos from 600 action categories, and Kinetics-700 [Carreira et al., 2019], which consists of 650,000 videos from 700 action categories, are currently available.

In this study, we used the subset of the Kinetics dataset named Kinetics-100 [Jing et al., 2021] because the original datasets, such as Kinetics-400, Kinetics-600, and Kinetics-700, are too massive to optimize deep networks on our available computational resources.

2.3 Compressed Video Action Recognition

Data compression is the process that encodes information using fewer bits than the original representation by identifying and eliminating statistical redundancy or removing unimportant information. Today, most media, including images and videos, are stored as compressed representations for efficient storage and transmission.

Most deep networks receive the raw data decoded from the compressed representations. These networks require decoding to obtain their inputs, leading to inefficient inference. To overcome this problem, some recent studies directly used compressed representations as their deep network inputs. For images, Park and Johnson proposed to use JPEG representations as inputs of deep networks [Park and Johnson, 2023]. While their model is only available for JPEG, ByteFormer is more general and directly processes bytes of compressed data by the large language models [Horton et al., 2023]. SeiT uses the codes of VQ-VAE as the compressed inputs instead of the popular data compression algorithms [Park et al., 2023]. Compressed vision also used codes of VQ-VAE as the inputs of deep networks for video understanding [Wiles et al., 2022].

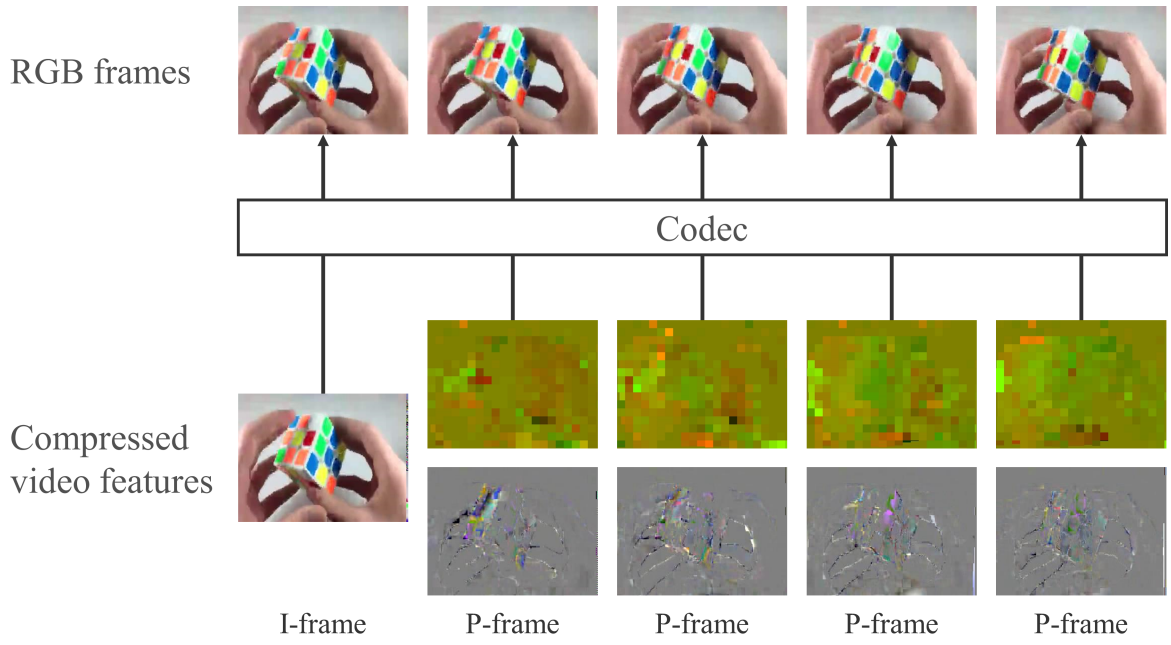


Fig. 2.5 Compressed video features.

Compressed video action recognition is one of the pioneering computer vision fields based on compressed data. The compressed video action recognition studies have mainly used compressed representations of MPEG4 [Le Gall, 1991] as inputs and performed video classification [Zhang et al., 2016, Zhang et al., 2018, Wu et al., 2018, Li et al., 2021]. Some studies extended compressed video action recognition into various applications for real-time object tracking [Bommes et al., 2020, Wang et al., 2019, Fields, 2019] and facial expression recognition [Liu et al., 2021a].

This study also focuses on compressed video action recognition. In this section describes inputs and previous studies of compressed video action recognition in more detail.

2.3.1 Compressed Video Features

Video is a sequence of RGB frames. In general, videos gradually change frame by frame, and their contents tend to be redundant among the consecutive frames. Highly efficient video compression algorithms such as MPEG-4, H.264, and HEVC leverage this fact and compress most frames by reusing contents from other frames and storing only the difference.

Most modern codecs arrange RGB frames into groups of pictures (GOPs). Each GOP starts with one intra-coded frame (I-frame), followed by predictive frames (P-frames), and zero or more bi-directional frames (B-frames). We depicted compressed video features in Fig. 2.5. I-frames are compressed as the standard images. P-frames reference the contents of the previous frames and only encode the differences between P-frames and the previous frames. More specifically, the differences are represented by motion vectors and residuals. The motion vectors represent movements of macroblocks of pixels from previous to current frames. After the compensation for macroblock movements, there can still be differences between the original and predicted frames. To correct the predicted frames after movement compensation, residuals store the pixel-wise differences in the RGB values between the original and predicted frames. B-frames are also represented by motion vectors and residuals similar to P-frames but reference previous and future frames. Then, the codecs transform motion vectors and residuals using discrete cosine transform (DCT) and entropy encoding for compression.

Compressed video action recognition classifies videos from I-frames, motion vectors, and residuals. This study compresses videos by the MPEG-4 Part2 format [Le Gall, 1991] and only uses I-frames and P-frames as inputs, following the previous study [Wu et al., 2018].

Back-tracing technique

A P-frame depends on the reference frame, which might also be a P-frame; this chain of dependencies continues back to the I-frame within the same GOP. Treating each P-frame as an independent observation ignores this dependency. To overcome this problem, Wu et al. proposed the back-tracing technique to decouple individual P-frames, making them dependent on only their predescent I-frames.

Let $\{I^{(0)}, I^{(1)}, ..., I^{(T)}\}$ be a GOP consists of an I-frame $I^{(0)}$ and T P-frames $\{I^{(1)}, ..., I^{(T)}\}$. Each P-frame $I^{(t)}$ is transformed to motion vectors $\mathcal{T}^{(t)}$ and residuals $\Delta_i^{(t)}$ for the compression, which satisfy

$$I_i^{(t)} = I_{i-\mathcal{T}_i^{(t)}}^{(t-1)} + \Delta_i^{(t)}, \tag{2.6}$$

for all pixel locations i. The back-tracing technique removes the dependency between consecutive P-frames by recursively tracing to an I-frame I^0. Formally, consider two frames in the same GOP: $I^{(k)}$ and $I^{(t)}$, where $k < t$. Then, the pixel location i of $I^{(t)}$ can be traced back to the corresponding location $\mathcal{J}_i^{(t,k)}$ of $I^{(k)}$ defined as follows:

$$\mu_{\mathcal{T}^{(t)}}(i) := i - \mathcal{T}_i^{(t)}, \tag{2.7}$$

$$\mathcal{J}_i^{(t,k)} := \mu_{\mathcal{T}^{(k+1)}} \circ \cdots \circ \mu_{\mathcal{T}^{(t)}}(i). \tag{2.8}$$

Now, we can accumulate motion vectors and residuals from $I^{(k)}$ to $I^{(t)}$ as follows:

$$\mathcal{D}_i^{(t,k)} := i - \mathcal{J}_i^{(t,k)}, \tag{2.9}$$

$$\mathcal{R}_i^{(t,k)} := \sum_{j=k+1}^{t} \Delta_{\mathcal{J}_i^{(t,j)}}^{j}, \tag{2.10}$$

where $\mathcal{D}_i^{(t,k)}$ and $\mathcal{R}_i^{(t,k)}$ denote accumulated motion vectors and residuals, respectively. Here, resulting accumulated motion vector $\mathcal{D}_i^{(t,k)}$ and residual $\mathcal{R}_i^{(t,k)}$ are independent on the P-frames in between $I^{(k)}$ and $I^{(t)}$. Therefore, when $k = 0$, the accumulated motion vector and residual depend on only the target P-frame $I^{(t)}$ and the I-frame $I^{(0)}$ within the same GOP. We depict the accumulated motion vectors and residuals in Fig. 2.6

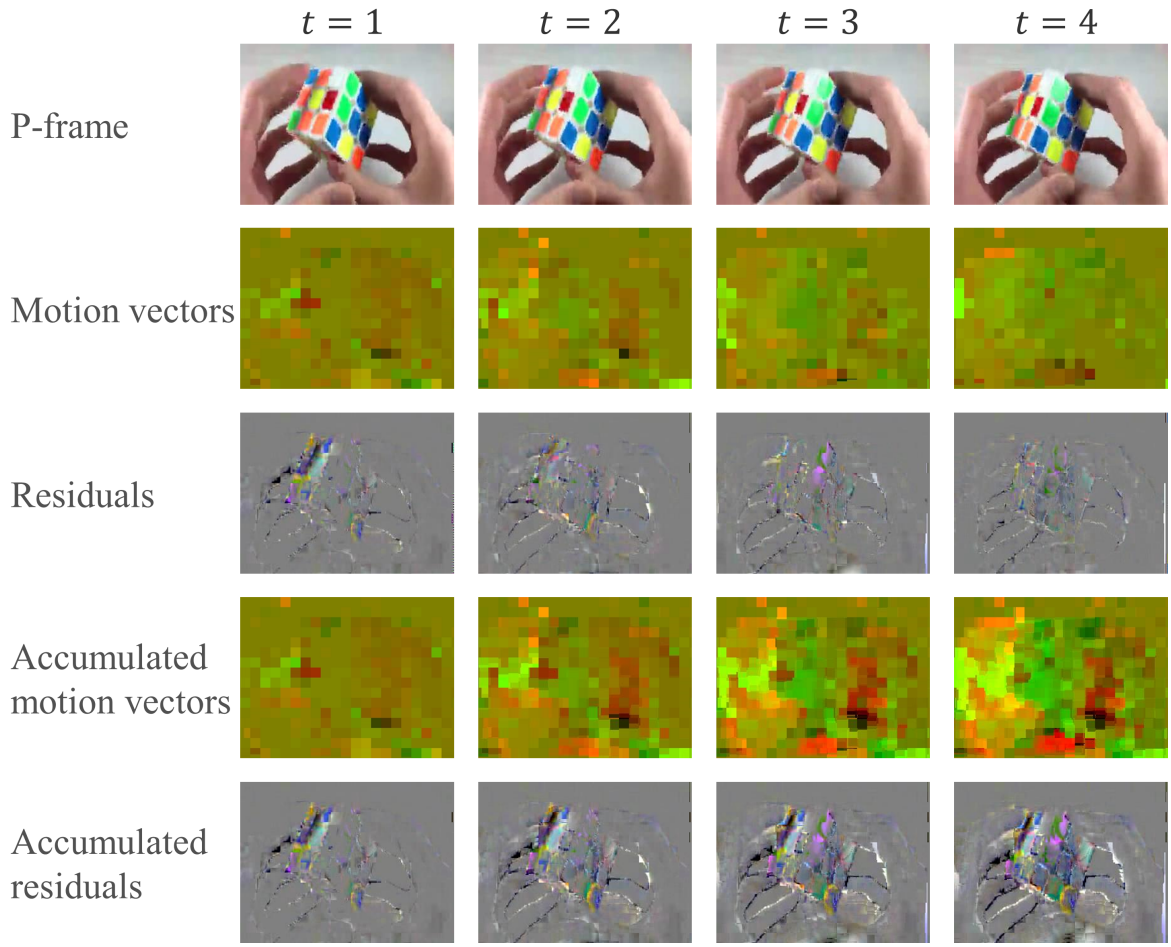


Fig. 2.6 Accumulated compressed video features using the back-tracing technique.

Wu et al. also claim that the accumulated motion vectors and residuals are more robust to noise and camera motion than the original ones because they contain longer-time information. Because of the advantage, this study always uses the accumulated motion vectors and residuals as model inputs without any mentions.

2.3.2 Model

Pioneering studies [Zhang et al., 2016, Zhang et al., 2018] on compressed video action recognition only used motion vectors as an easy-to-use alternative to optical

flow that needs expensive computation to obtain. These studies used decoded RGB frames and did not use compressed video features other than motion vectors; thus, their efficiency was still limited.

Wu et al. [Wu et al., 2018] first proposed the CoViAR method that classifies videos using only compressed video features. They employed three 2DCNNs corresponding to compressed video features and trained them independently. After training, the final prediction was computed by averaging the predictions of the three different networks. Wu et al. also introduced the back-tracing technique to improve compressed video action recognition. Li et al. [Li et al., 2021] showed that compressed video action recognition was available under the practical scenario that compressed videos are transmitted from other devices, and some packets are dropped.

Subsequent studies have developed more efficient or effective compressed video action recognition methods by replacing backbone networks with different lightweight networks and employing additional components to maintain the accuracy of the CoViAR method. For example, CV-C3D [dos Santos et al., 2019] and MFCD-Net [Battash et al., 2020] used 3DCNNs; Wu et al. [Wu et al., 2019] and Guo et al. [Guo et al., 2023] used ResNet18 [He et al., 2016] as their backbone network and trained it using knowledge distillation [Hinton et al., 2015]; TTP [Huo et al., 2019] combined MobileNetV2 [Sandler et al., 2018] with an efficient yet effective fusion method. Other studies estimated optical flow from motion vectors and residuals and used the estimated optical flow to improve accuracy. The DMC-Net method [Shou et al., 2019] trained the optical flow estimator in a supervised manner using actual optical flow for training. The subsequent SIFP method [Li et al., 2020] developed an unsupervised approach to train the optical flow estimator without actual optical flow. The proposed MussNet provides another approach for efficient compressed video action recognition.

Chapter 3

Semi-supervised Compressed Video Action Recognition to Reduce Annotation Cost

3.1 Introduction

As described in Chapter 2, deep networks have shown remarkable progress in video classification [Hara et al., 2017, Tran et al., 2018, Feichtenhofer, 2020, Feichtenhofer et al., 2019]. The deep networks capture the spatiotemporal features of videos using their uncountable trainable parameters. However, to optimize their parameters, we require large-scale annotated datasets [Soomro et al., 2012, Kuehne et al., 2011, Carreira and Zisserman, 2017] to avoid over-fitting. This is a critical problem because annotating many videos manually is expensive and tedious, and preparing such datasets is challenging. For example, the recent massive Ego4D dataset, which collects 3,670 hours of video, consumes over 250,000 hours of annotator effort for annotation [Grauman et al., 2022].

Some studies have focused on semi-supervised learning (SSL) to reduce the depen-

dencies of annotations [Jing et al., 2021, Zou et al., 2021, Xiong et al., 2021]. SSL uses both labeled and unlabeled data to train deep networks. Effective SSL methods utilize unlabeled data efficiently and obtain robust models even if labeled data are limited. In SSL, "pseudo-labeling" is a popular approach that generates artificial labels from the model predictions for unlabeled data. In this approach, how artificial labels are generated significantly impacts performance. Xiong et al. proposed multiview pseudo labeling (MvPL) [Xiong et al., 2021] as a semi-supervised video classification method based on pseudo-labeling. MvPL generates accurate artificial labels by ensembling the model predictions from the RGB frames, optical flow, and temporal gradients. An optical flow represents the apparent motion of the objects, and temporal gradients represent the differences in RGB between two frames. Because they emphasize temporal features, these representations help models efficiently learn spatio-temporal features.

We consider that conventional SSL methods have the following problems. First, raw RGB videos have many redundancies that are irrelevant to classify videos. In general, most videos change frame by frame gradually so that much information is also contained in the neighbor frames and redundant. Under the limited labels, the training models would be confused by the redundancies and suffer from serious over-fitting. Second, conventional SSL methods require computational and storage costs to obtain their model's inputs, making it difficult to scale up the training. Although these SSL methods use RGB frames as their model inputs, we often store videos as compressed video files to reduce the file size. Therefore, to extract RGB frames, we must decode them from compressed video files every time they are used. This decoding process increases the computational cost and makes the training inefficient. In addition, using an optical flow, such as MvPL, requires significant computational cost. To reduce the computational costs during training, we often compute RGB frames (along with the optical flow) in advance and save them as images. However, this approach consumes

a significant amount of time for preprocessing and requires a large storage space. Hence, the previous SSL methods require high-end computer clusters to train deep networks on large-scale datasets. In particular, with MvPL, which uses RGB frames and optical flow for training, it is more challenging to use large-scale datasets. This is a critical problem for SSL because some studies have shown that increasing unlabeled data improves its performance [Oliver et al., 2018, Yalniz et al., 2019]; however, high computational and storage costs hinder the utilization of many unlabeled data for training.

To address this problem, we combine SSL with compressed video action recognition. As described in Chapter 2, compressed videos contain multiple types of features converted from RGB frames for efficient storage, and we can directly load them without any decoding. This means that using compressed videos as inputs requires neither the computational costs for decoding nor storage costs for maintaining the RGB frames (and the optical flow) as images, making it easier for SSL to scale up the training. Furthermore, some features stored in compressed videos hold temporal features similar to optical flow and temporal gradients. Therefore, we can employ the ensemble approach for pseudo-labeling. Although compressed videos are already used for video classification [Wu et al., 2018, Shou et al., 2019], this is the first study that uses compressed videos for semi-supervised video classification.

The remainder of this chapter is organized as follows. In Section 3.2, we introduce previous SSL studies. In Section 3.3, we define notations for the following sections. In Section 3.4, we apply Lee's method [Lee et al., 2013] to compressed video action recognition and show that compressed video features are preferable to RGB frames under the limited labels for accuracy. In Section 3.5, we propose Compressed Video Ensemble based Pseudo Labeling (CoVEnPL) for semi-supervised compressed video action recognition. Finally, we conclude this chapter in Section 3.6.

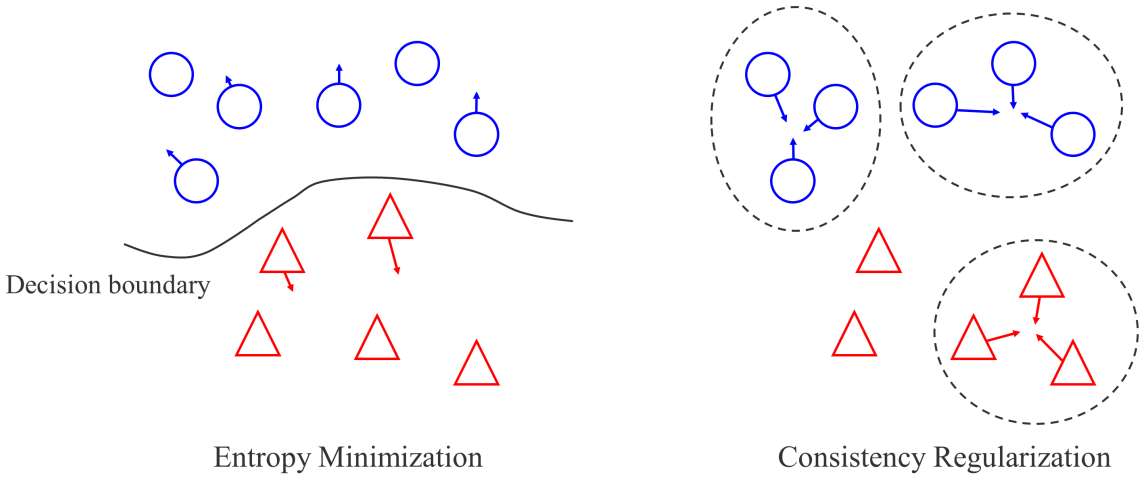


Fig. 3.1 Abstract image of entropy minimization and consistency regularization. The blue circles represent examples of one class, and the red triangles represent those of another class.

3.2 Background: Semi-supervised Learning

3.2.1 Semi-supervised Learning for Images

Most previous studies on SSL in computer vision have focused on image classification. In SSL, "pseudo-labeling" is a popular approach. This approach is based on the idea that we can generate artificial labels from the model predictions and use the generated artificial labels for supervised learning of unlabeled data. To design how to generate artificial labels, there are two critical factors: entropy minimization and consistency regularization, and most SSL algorithms contain either or both of the factors.

Entropy minimization

This regularization aims to minimize the entropy of model predictions. In other words, entropy minimization encourages models to output high-confidence predictions. In classification problems, the decision boundary should not pass through high-

density regions of training data within the learned latent space. This idea is widely accepted in machine learning methods such as a support vector machine [Vapnik, 1963]. Entropy minimization helps models learn such latent spaces and the decision boundary from unlabeled data. The early SSL method based on entropy minimization was proposed by Lee [Lee et al., 2013]. This method first generates the model predictions from unlabeled data and creates one-hot labels from such predictions as artificial labels of unlabeled data. Lee's method uses only high-confidence predictions for generating artificial labels to avoid inaccurate artificial labels. Then, the artificial labels are used as the target labels for unlabeled data to train the models in a supervised manner. As other methods for entropy minimization, a sharpen function [Berthelot et al., 2019] and a loss term that explicitly minimizes the entropy of predictions [Miyato et al., 2018] have been proposed. The sharpen function sharpens the model predictions for generating artificial labels and will be integrated into the proposed method.

Consistency regularization

This regularization aims to obtain robust models against any perturbations. Specifically, this regularization generates two different predictions from the same unlabeled data by adding different perturbations and minimizing the distance or divergence between the predictions. Π-Model [Laine and Aila, 2016] is a typical consistency regularization method that applies small perturbations such as a dropout, random cropping, and horizontal flipping to unlabeled data. Recent studies have used both small perturbations and large perturbations. The large perturbations make the input data challenging to classify correctly; therefore, maintaining consistent predictions becomes more difficult for the applied models. More robust models than the Π-model are obtained by training using such perturbations. Specifically, virtual adversarial training [Miyato et al., 2018] uses adversarial perturbations, and unsupervised data augmentation [Xie et al., 2020] uses RandAugment [Cubuk et al., 2020] as the large

perturbations. In the consistency regularization, either prediction from unlabeled data is considered an artificial label. Some consistency regularization methods aim to make this artificial label more stable and accurate by generating artificial labels from the ensemble of previous networks. Temporal Ensembling [Laine and Aila, 2016] uses the exponential moving average (EMA) of the past predictions of each unlabeled data as its artificial label, and Mean Teacher [Tarvainen and Valpola, 2017] uses a teacher network obtained from the EMA of past network weights to generate artificial labels.

The recent trend of SSL is combining entropy minimization and consistency regularization. In particular, FixMatch [Sohn et al., 2020] combines Lee's method and RandAugment to achieve a state-of-the-art performance.

3.2.2 Semi-supervised Learning for Videos

Compared to image classification, only a few studies have focused on semi-supervised video classification. Jing et al. proposed combining Lee's method with distillation [Girdhar et al., 2019] from a pre-trained image classifier to help with video classification training [Jing et al., 2021]. Zou et al. explored a set of suitable transformations of RandAugment for videos and used them with FixMatch for SSL. These methods use only RGB frames to train the models, limiting their performances.

Some works proposed semi-supervised video classification with multi-stream input, and various types of inputs are explored in their methods. MvPL [Xiong et al., 2021] is a recent state-of-art method for semi-supervised video classification. This method generates artificial labels by ensembling the predictions from RGB frames, optical flows, and temporal gradients. Because an ensemble of the predictions from multiple types of features boosts the performances [Simonyan and Zisserman, 2014, **?**], MvPL can obtain reliable artificial labels and achieve a promising performance. In addition, MvPL trains a single model to classify videos from all features; thus, MvPL can classify videos from only RGB frames and reduce the computational cost of optical

flows and temporal gradients during the inference phase. Inspired by MvPL, Xiao et al. proposed the SSL method that does not use optical flow [Xiao et al., 2021]. They used temporal gradients computed from RGB frames of different FPS videos instead of optical flow. Although their method does not outperform MvPL, it is much better than other methods that use only RGB frames as inputs. CMPL also uses multi-stream inputs [Xu et al., 2021] although it does not employ the ensembling approach. CMPL uses RGB frames of different FPS videos and achieves competitive performances against MvPL. Like these methods, our proposed method employs multi-stream input and ensembles the predictions to generate reliable artificial labels.

These conventional semi-supervised video classification methods overlook the computational and storage costs required to obtain their input features because we often store videos as compressed video files. This problem makes the training of conventional SSL methods inefficient. In particular, MvPL training is highly inefficient because it uses an optical flow, and acquiring it may incur significant computational costs. We address this problem by using directly available features from compressed videos as our model inputs.

3.3 Preliminaries

Let $\mathcal{X}$ be a set of labeled videos and $\mathcal{U}$ be a set of unlabeled videos. For semi-supervised learning, we sample a subset of labeled videos from $\mathcal{X}$ and that of unlabeled videos from $\mathcal{U}$ as mini-batches. The subset of labeled videos is represented as $B_{sup} = \{(x_i, t_i)|1 \leq i \leq N_{sup}\}$, where x_i is an i-th video, t_i is a label of x_i and N_{sup} is the number of labeled videos and the subset of unlabeled videos is represented as $B_{unsup} = \{u_j|1 \leq j \leq N_{unsup}\}$, where u_j is a j-th video and N_{unsup} is the number of unlabeled videos.

We sample T GOPs from videos as inputs, where one GOP holds one I-frame and multiple P-frames. Then, we subsample μ P-frames from each GOP while maintaining

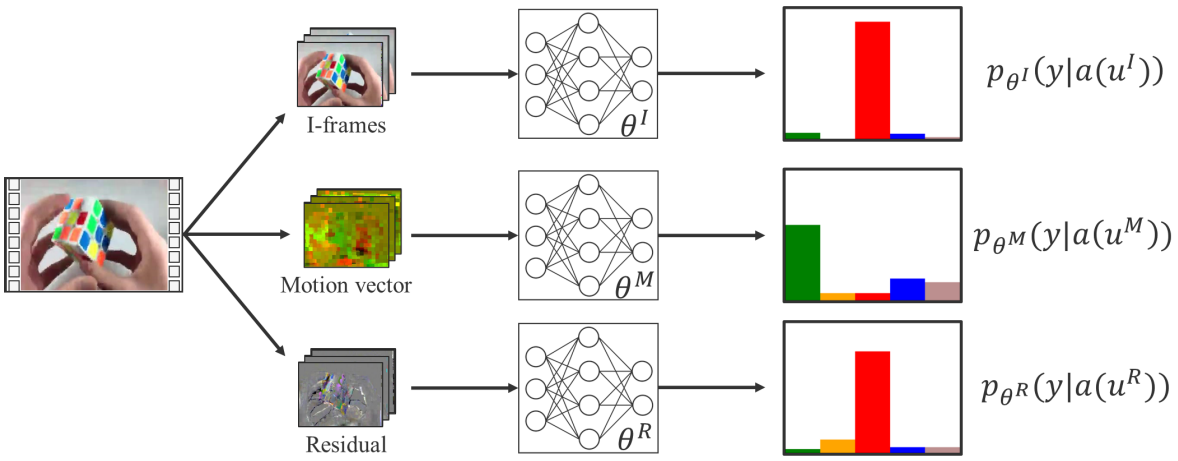


Fig. 3.2 The abstract image of the multi-stream model used in this chapter.

the frame order. Consequently, we can obtain T I-frames and μT P-frames. For simplicity, x_i^I, x_i^M, and x_i^R denote I-frames, motion vectors and residuals of x_i, and u_j^I, u_j^M, and u_j^R denote I-frames, motion vectors and residuals of u_j.

To process compressed video features, we use a multi-stream model that consists of three independent networks corresponding to I-frames, motion vectors, and residuals (Fig. 3.2). Each network receives the corresponding input and returns the probability distribution for classification. Formally, given $\{x_i^I, x_i^M, x_i^R\}$ as inputs, the predicted distributions are denoted as $\{p_{\theta^I}(y_i|x_i^I), p_{\theta^M}(y_i|x_i^M), p_{\theta^R}(y_i|x_i^R)\}$, where θ^I, θ^M and θ^R are network parameters of I-frames, motion vectors and residuals, respectively.

The predicted distributions should be fused during inference to obtain the final prediction. In this study, the final prediction is computed by averaging the three predicted distributions as follows:

$$\frac{1}{3}\sum_{V\in\{I,M,R\}} p_{\theta^V}(y_i|x_i^V). \tag{3.1}$$

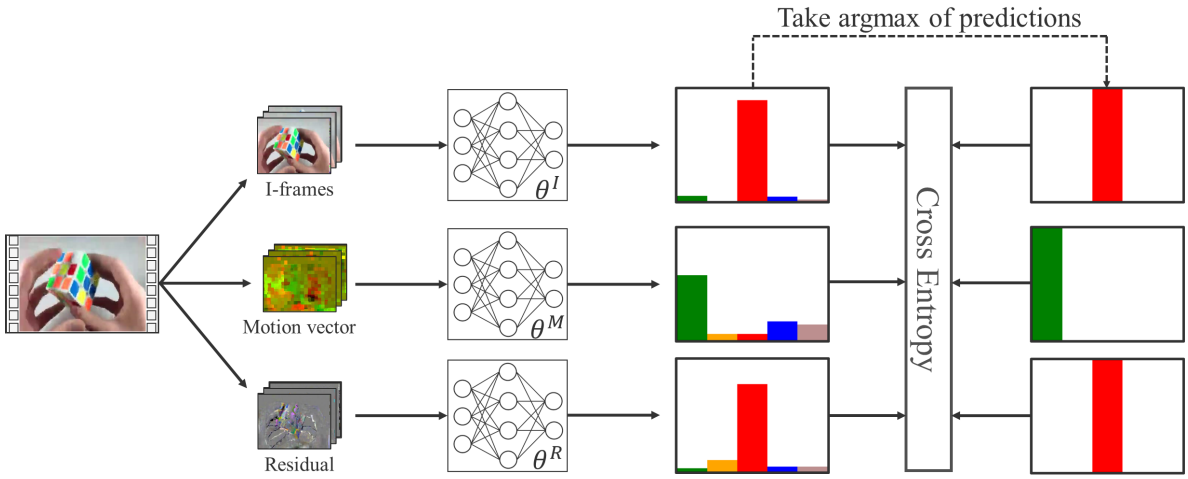


Fig. 3.3 Pseudo labeling method for semi-supervised compressed video action recognition.

3.4 Pseudo Labeling for Semi-supervised Compressed Video Action Recognition

3.4.1 Training Method

We first train the networks independently using the same objective function as Lee [Lee et al., 2013]. This objective function separately computes supervised loss $\mathcal{L}_{sup}$ from labeled data and unsupervised loss $\mathcal{L}_{unsup}$ from unlabeled data and combines them for the total loss.

For labeled data B_{sup}, we compute supervised loss $\mathcal{L}_{sup}$ defined as:

$$\mathcal{L}_{sup} = \frac{1}{3N_{sup}} \sum_{i=1}^{N_{sup}} \sum_{V \in \{I,M,R\}} H(p_{\theta^V}(y|a(x_i^V)), t_i), \tag{3.2}$$

where H is the cross-entropy loss as Section 2.1 and a is stochastic augmentation that transforms x. Thanks to the compressed video features, the networks can learn efficiently under the limited number of labeled data via supervised loss.

For unlabeled data B_{unsup}, we compute artificial labels from unlabeled inputs u_j^I, u_j^M, and u_j^R. To obtain artificial labels, we first compute the model predictions from

I-frames, motion vectors, and residuals as follows:

$$q_j^I = p_{\theta^I}(y|a(u_j^I)), \tag{3.3}$$
$$q_j^M = p_{\theta^M}(y|a(u_j^M)), \tag{3.4}$$
$$q_j^R = p_{\theta^R}(y|a(u_j^R)). \tag{3.5}$$

From these predictions, the artificial labels can be obtained as follows:

$$\hat{q}_j^I = \arg\max(q_j^I), \tag{3.6}$$
$$\hat{q}_j^M = \arg\max(q_j^M), \tag{3.7}$$
$$\hat{q}_j^R = \arg\max(q_j^R). \tag{3.8}$$

Now, we define the unsupervised loss $\mathcal{L}_{unsup}$ as the sum of the cross-entropy loss between the model predictions $\{q_j^I, q_j^M, q_j^R\}$ and the artificial labels $\{\hat{q}_j^I, \hat{q}_j^M \hat{q}_j^R\}$. as follows:

$$\mathcal{L}_{unsup} = \frac{1}{3N_{unsup}} \sum_{j=1}^{N_{unsup}} \sum_{V \in \{I,M,R\}} \mathbb{1}(\max(q_j^V) > \texttt{threshold}) H(q_j^V, \hat{q}_j^V), \tag{3.9}$$

where $\mathbb{1}(\max(q_j^V) > \texttt{threshold})$ returns 1 if the prediction confidence $\max(q_j^V)$ is higher than `threshold`; otherwise, returns 0. This term is introduced to avoid unreliable artificial labels during training. The unsupervised loss $\mathcal{L}_{unsup}$ encourages our models to make high-confidence predictions. In other words, it works as entropy minimization mentioned in Section 3.2.1.

Finally, we combine the supervised loss $\mathcal{L}_{sup}$ and the unsupervised loss $\mathcal{L}_{unsup}$ and define our objective function as follows:

$$\mathcal{L} = \mathcal{L}_{sup} + \lambda \mathcal{L}_{unsup}, \tag{3.10}$$

where λ is the hyperparameter determining the relative weight of an unsupervised loss $\mathcal{L}_{unsup}$. The general algorithm of the proposed method is summarized in Algorithm 1.

Algorithm 1 The extended version of Lee's method into semi-supervised compressed video action recognition.

Require: Network parameters $\{\theta^I, \theta^M, \theta^R\}$, labeled data $\mathcal{X}$, unlabeled data $\mathcal{U}$, threshold `threshold`, balancing weight λ, learning rate η

Ensure: Updated parameters $\{\theta^I, \theta^M, \theta^R\}$

while $\{\theta^I, \theta^M, \theta^R\}$ do not converge **do**
  // Compute supervised loss
  $B_{sup} = \{(x_i, y_i) | 1 \leq i \leq N_{sup}\} \sim \mathcal{X}$
  Get $\{(x_i^I, x_i^M, x_i^R) | 1 \leq i \leq N_{sup}\}$ from B_{sup}.
  $\mathcal{L}_{sup} = \frac{1}{3N_{sup}} \sum_{(x_i,t_i)\in B_{sup}} \sum_{V\in\{I,M,R\}} H(p_{\theta^V}(y_i^V | a(x_i^V)), t_i)$
  // Compute unsupervised loss
  $B_{unsup} = \{u_j | 1 \leq j \leq N_{unsup}\} \sim \mathcal{U}$
  Get $\{(u_j^I, u_j^M, u_j^R) | 1 \leq j \leq B_u\}$ from B_{unsup}.
  for V **in** $\{I, M, R\}$ **do**
    $q^V \leftarrow p_{\theta^V}(y | a(u^V))$
    $\hat{q}^V \leftarrow \arg\max q^V$
  end for
  $\mathcal{L}_{unsup} = \frac{1}{3N_{unsup}} \sum_{j=1}^{N_{unsup}} \sum_{V\in\{I,M,R\}} \mathbb{1}(\max(q_j^V) > \texttt{threshold}) H(q_j^V, \hat{q}_j^V)$
  // Compute loss to minimize
  $\mathcal{L} = \mathcal{L}_{sup} + \lambda \mathcal{L}_{unsup}$
  // Update parameters by SGD
  $\theta^I \leftarrow \theta^I - \eta \frac{\partial \mathcal{L}}{\partial \theta^I}$
  $\theta^M \leftarrow \theta^M - \eta \frac{\partial \mathcal{L}}{\partial \theta^M}$
  $\theta^R \leftarrow \theta^R - \eta \frac{\partial \mathcal{L}}{\partial \theta^R}$
end while

3.4.2 Experimental Setup

We use UCF-101 to evaluate the proposed method. All videos are encoded as MPEG-4 Part2 [Le Gall, 1991], where each GOP holds one I-frame and 11 P-frames, and we resized all videos to 171 × 128 resolution to reduce the computational cost. During the training, we performed data augmentations by randomly cropping 112 × 112 patches from input frames and flipping the cropped patches horizontally with a 50% probability. For evaluation, we cropped 112 × 112 patches by center cropping. The size of each frame was 112 pixels × 112 pixels × 3 channels for I-frames and residuals and 112 pixels × 112 pixels × 2 channels for motion vectors, respectively.

Implementation details

Our multi-stream model consists of three ResNet-18 [He et al., 2016] implemented in torchvision [*1]. We trained these networks independently and combined them for inference. As hyperparameters, we always use $\lambda = 1$, and $\texttt{threshold} = 0.95$. Each network is updated 2^{18} times using SGD with Nesterov's momentum [Sutskever et al., 2013]. We set the learning rate to 0.01, momentum to 0.9, and weight decay to 0.0005. For temporal augmentation, we set $T = 3$ and $\mu = 1$ to select frames, and thus we input 3 I-frames and a P-frame into our networks. Each batch for training contained 64 labeled and 64 unlabeled data. We trained our model from scratch on the different percentages of labeled data. To train our model in a semi-supervised manner, we randomly sampled 5%, 10%, 20%, or 50% of the dataset as labeled data and used the remains as unlabeled data. Unlabeled data was annotated, but we discarded their labels to train models in a semi-supervised manner. We ran the training three times under the same settings and reported the average accuracy scores.

[*1] https://pytorch.org/docs/1.5.0/torchvision/models.html

Baseline methods

As a supervised baseline model, we trained our model with only supervised loss $\mathcal{L}_{sup}$, where unlabeled data were not used, while the other settings were the same as our proposed semi-supervised methods described in Section 4.1 and 4.2. We call this model Supervised Learning on Compressed Video Representations. To evaluate the effectiveness of the compressed video representations in supervised learning, we compared our results with the results reported by Jing et al. [27], which trained 3D ResNet18 [3] on raw videos. Specifically, as supervised baselines on raw videos, we cited results given by the following scenarios:

- Supervised learning: Training 3D ResNet18 on only labeled raw videos. This model is trained to minimize supervised loss $\mathcal{L}_{sup}$.
- Supervised learning with knowledge distillation: Training 3D ResNet18 on only labeled raw videos. This model utilizes knowledge distillation using an extra image dataset; it is trained to minimize supervised loss $\mathcal{L}_{sup}$ and to mimic the outputs of the ImageNet pre-trained classifier. This technique is a core component of VideoSSL, as mentioned below.

For semi-supervised baselines on raw videos, we cited the results given by the following methods:

- Pseudo Label: Training 3D ResNet18 by Lee's method [Lee et al., 2013] on raw videos. This method uses the same objective function as our method.
- Mean Teacher: Training 3D ResNet18 by Mean Teacher [Tarvainen and Valpola, 2017] on raw videos.
- S4L: Training 3D ResNet18 by Self-Supervised Semi-Supervised Learning (S4L) method [Zhai et al., 2019]. In S4L, the model is trained by the standard classification task on labeled data and the self-supervised task of predicting the spatial rotation of images on unlabeled data.

Table 3.1 Accuracy comparison of supervised learning on the UCF-101 dataset. Accuracy scores were achieved, where different percentages of data were used as labeled data, and the remainder was used as unlabeled data.

Method	5%	10%	20%	50%
Supervised learning	16.9	24.0	32.2	38.3
Supervised learning with knowledge distillation	31.2	40.7	45.4	53.9
Supervised learning on compressed video features	36.2	45.7	56.7	66.2

- VideoSSL: Combining Lee's method [Lee et al., 2013] and *Supervised Learning with Knowledge Distillation*.

3.4.3 Comparison with Supervised Learning

Table 3.1 compares the training results in a supervised manner. As shown in Table 3.1, our method (supervised learning on compressed video features) consistently outperformed *Supervised learning* and *Supervised learning with knowledge distillation*, both of which were trained on raw videos. These results showed that our method prevented over-fitting under the limited number of labels by reducing redundancies with video compression. The comparison between our method and *Supervised learning with knowledge distillation* also showed that training on compressed video representations is more efficient than using an extra dataset like ImageNet.

3.4.4 Comparison with Semi-supervised Learning

Table 3.2 compares our method with semi-supervised learning methods trained with 3D ResNet18 on raw videos. Similar to the results of supervised learning, our proposed method consistently outperformed other methods. For example, when trained models on 10% labels of UCF101, our proposed method achieved 47.9% accuracy, which is

Table 3.2 Accuracy comparison of semi-supervised learning on the UCF-101 dataset. Accuracy scores were achieved, where different percentages of data were used as labeled data, and the remainder was used as unlabeled data.

Method	5%	10%	20%	50%
Pseudo Label	17.6	24.7	37.0	47.5
Mean Teacher	17.5	25.6	36.3	45.8
S4L	22.7	29.1	37.7	47.9
VideoSSL	32.4	42.0	48.7	54.3
Proposed	**36.8**	**47.9**	**58.1**	**66.6**

25.2 points higher than Pseudo Label and 5.9 points higher than VideoSSL. Even when compared to our proposed method trained on 20% labels with VideoSSL trained on 50% labels, our method outperformed VideoSSL by 3.8 points. These results showed that our method is more effective than training on raw videos for semi-supervised video classification.

3.4.5 Discussion

Our experiments show that our method is more effective than training on raw videos for both supervised and semi-supervised learning. This would be because raw videos have many redundancies that are irrelevant to classifying videos. Under the limited labels, the training models would be confused by the redundancies and suffer from serious over-fitting. Our method can relax such over-fitting by the video compression.

Our proposed semi-supervised learning method is better than VideoSSL, which is the best semi-supervised learning method among the previous methods. Furthermore, our method has another advantage against VideoSSL: it is more robust toward the various target domains. VideoSSL can improve performance on the UCF101 dataset because the ImageNet pre-trained classifier gives training models a hint of correct

action labels via class distributions of ImageNet, whether the input is labeled or not. For example, when given unlabeled data of "Soccer Juggling," the ImageNet pre-trained classifier will give models the class distributions with high confidence on "soccer ball." In the case of the UCF101 dataset, this signal is beneficial for training models because it contains actions of "Soccer Juggling" and "Soccer Penalty," which are related to "soccer ball." However, if we use VideoSSL for surveillance to monitor criminal activities, the ImageNet pre-trained classifier cannot give helpful signals about criminal activities to training models. This is because most classes related to humans in ImageNet are about clothes such as sunglasses, trench coats, or suits, which are irrelevant to criminal activities. In such cases, the improvement of VideoSSL is limited. On the other hand, our method never suffers from this problem because our method does not rely on pre-training with extra datasets such as ImageNet. Our method applies to a broader range of domains for video classification.

3.5 Compressed Video Ensemble-based Pseudo Labeling

We propose CoVEnPL, which trains models using compressed videos in a semi-supervised manner. We show an overview of CoVEnPL training for unlabeled data in Fig. 3.4.

CoVEnPL is a pseudo-labeling approach and generates artificial labels from the ensemble of predictions from multiview inputs such as MvPL. The main difference between CoVEnPL and MvPL is that CoVEnPL uses features stored in compressed videos as inputs to reduce computational and storage costs. The generated artificial labels are used to train the models through a variant version of FixMatch, one of the best SSL methods.

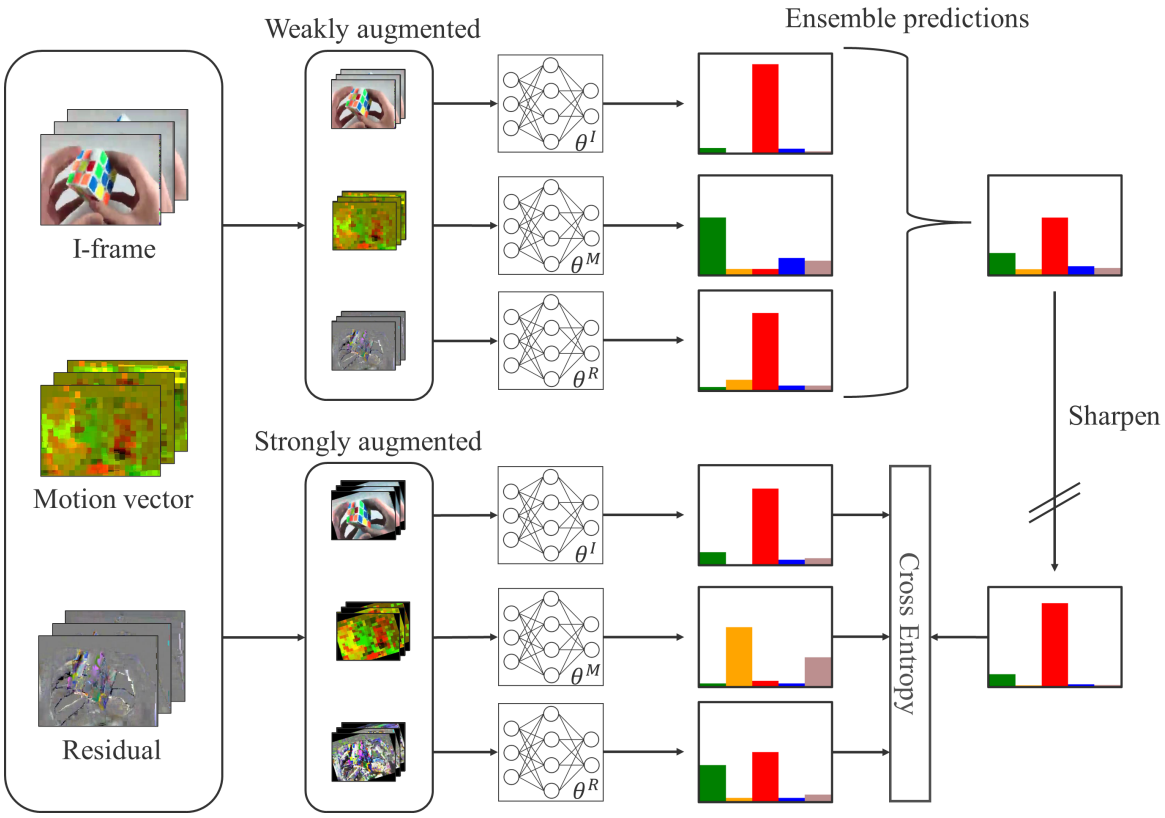


Fig. 3.4 Overview of CoVenPL method.

3.5.1 Augmentation for Strong Perturbations

Data augmentation virtually increases the training data by using stochastic transformations and improves the robustness of the models. Recent SSL methods for image classification use weak and strong augmentation to add more significant perturbations for consistency regularization. Moreover, spatial and temporal augmentation is crucial for video classification because videos hold both features, and their capture is essential.

Following the temporal augmentation, we apply spatial augmentation to the extracted frames while maintaining the random parameters. CoVEnPL uses weak and strong spatial augmentation to control the perturbation strength. As a weak spa-

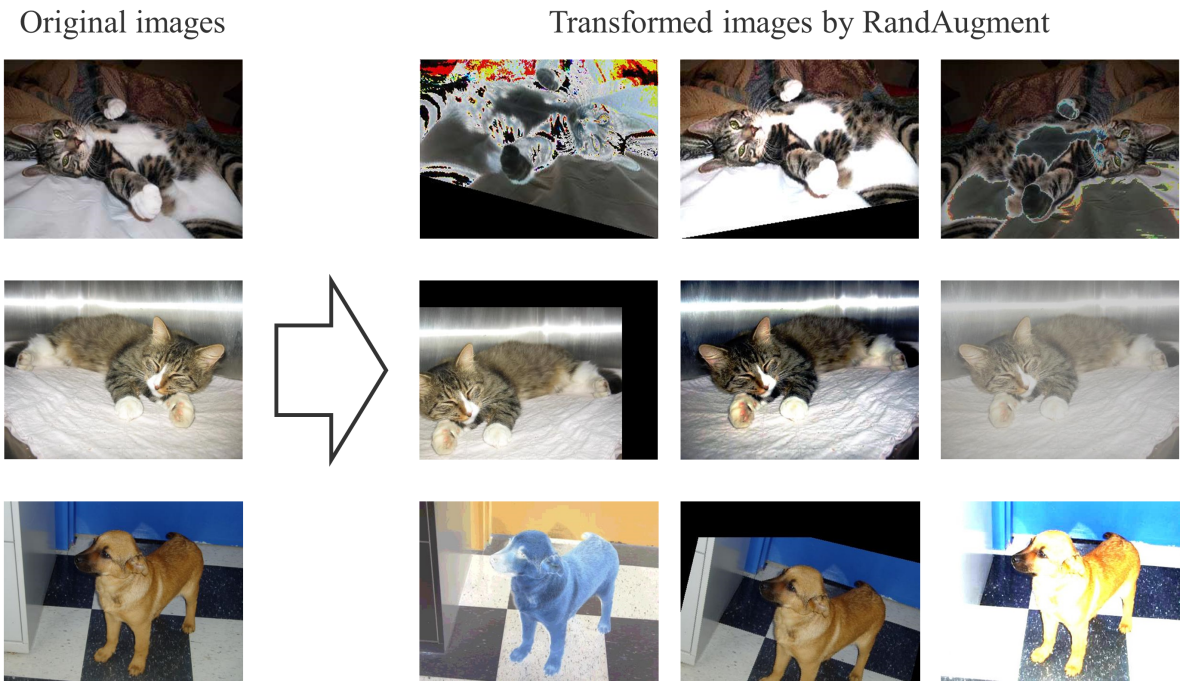


Fig. 3.5 Example of images transformed by RandAugment.

tial augmentation, we apply random cropping and horizontal flipping to the frames extracted through temporal augmentation. By contrast, as a strong spatial augmentation, we use RandAugment [Cubuk et al., 2020] following the previous SSL methods [Xie et al., 2020, Berthelot et al., 2020, Sohn et al., 2020, Xiong et al., 2021]. We show the general algorithm of RandAugment in Algorithm 2 and example images transformed by RandAugment in Fig. 3.5. RandAugment achieves a strong transformation of inputs by using randomly sampled multiple augmentations and their magnitude from the set of augmentations summarized in Table 3.3. Because there are no previous studies applying a strong augmentation to compressed videos, it is unclear what transformations are suitable for each feature of such a video, and it could be beneficial for the performance to search for the best set of transformations. However, instead of searching for the best transformations for compressed videos in this paper, we use RandAugment with the same transformations and their parameters

Algorithm 2 RandAugment algorithm used in CoVEnPL.

Require: Inputs $\{x^I, x^M, x^R\}$, number of transforms N, number of magnitude bins M

Ensure: Transformed inputs $\{x^I, x^M, x^R\}$

1: **for** $i = 1$ **to** N **do**
2: // Randomly sample transform and its parameter range from Table 3.3
3: $(\text{transform}, p_{min}, p_{max}) \sim transforms$
4: $m \sim \{1, 2, ..., M\}$
5: // Transform compressed video features.
6: $x^I \leftarrow \text{transform}(x^I, \frac{m(p_{max}-p_{min})}{M} + p_{min})$
7: $x^M \leftarrow \text{transform}(x^M, \frac{m(p_{max}-p_{min})}{M} + p_{min})$
8: $x^R \leftarrow \text{transform}(x^R, \frac{m(p_{max}-p_{min})}{M} + p_{min})$
9: **end for**

Ensure: x^I, x^M, x^R

applied in semi-supervised image classification [Sohn et al., 2020]. We found that using such transformations largely improves the performances of semi-supervised video classification despite being tuned for an image classification task.

3.5.2 Training Method

For labeled data B_{sup}, we also use the same supervised loss of Section 3.4, defined as:

$$\mathcal{L}_{sup} = \frac{1}{3N_{sup}} \sum_{i=1}^{N_{sup}} \sum_{V \in \{I,M,R\}} H(p_{\theta^V}(y|a(x_i^V)), t_i). \tag{3.11}$$

For unsupervised data B_{unsup}, we first make the ensemble prediction of unlabeled

Transformation	Description	Parameter	Range
Auto contrast	Maximizes the image contrast by setting the darkest (lightest) pixel to black (white).		
Brightness	Adjusts the brightness of the image. $B = 0$ returns a black image, $B = 1$ returns the original image.	B	[0.05, 0.95]
Color	Adjusts the color balance of the image like in a TV. $C = 0$ returns a black & white image, $C = 1$ returns the original image.	C	[0.05, 0.95]
Contrast	Controls the contrast of the image. A $C = 0$ returns a gray image, $C = 1$ returns the original image.	C	[0.05, 0.95]
Equalize	Equalizes the image histogram.		
Identity	Returns the original image.		
Posterize	Reduces each pixel to B bits.	B	[4, 8]
Rotate	Rotates the image by θ degrees.	θ	[-30, 30]
Sharpness	Adjusts the sharpness of the image, where $S = 0$ returns a blurred image, and $S = 1$ returns the original image.	S	[0.05, 0.95]
Shear_x	Shears the image along the horizontal axis with rate R.	R	[-0.3, 0.3]
Shear_y	Shears the image along the vertical axis with rate R.	R	[-0.3, 0.3]
Solarize	Inverts all pixels above a threshold value of T.	T	[0, 1]
Translate_x	Translates the image horizontally by ($\lambda \times$image width) pixels.	λ	[-0.3, 0.3]
Translate_y	Translates the image vertically by ($\lambda \times$image height) pixels.	λ	[-0.3, 0.3]

Table 3.3 List of transformations used by RandAugment, cited from Table 12 in [Sohn et al., 2020].

videos from predictions of the training networks as follows:

$$q_j = \frac{1}{3} \sum_{V \in \{I,M,R\}} p_{\theta^V}(y|a(u_j^V)). \tag{3.12}$$

We then generate artificial labels $\hat{q}$ from the estimated predictions as follows:

$$\hat{q}_j = \frac{q_j^{1/\tau}}{\sum_{k=1}^{C} q_{jk}^{1/\tau}}, \tag{3.13}$$

where q_{jk} denotes the k-th element of the ensembled prediction q_j, C is the number of classes, and τ is a sharpening temperature parameter determining how much the sharpen function sharpens the predictions. In the standard FixMatch, the artificial labels $\hat{q}$ are defined as one-hot labels from only the high-confidence predictions. However, we experimentally observe that most predictions from unlabeled data cannot achieve high confidence and are ignored. We consider this to occur because the three networks must predict the same action category to make the ensembled prediction achieve high confidence. One way to increase the unlabeled data utilization is to set the confidence threshold to a low value; however, Sohn et al. showed that using a low confidence threshold decreases the performance of FixMatch, and we should thus use a high confidence threshold. Hence, we decided to follow the variant version of FixMatch to address this problem. In this version, artificial labels $\hat{q}$ are generated from all estimated predictions q regardless of the confidence by applying the sharpen function. This means we can fully utilize unlabeled data for training without decreasing the final performance. Finally, we define the unsupervised loss $\mathcal{L}_{unsup}$ as the cross entropy loss between the sharpened artificial labels and the predictions from strongly augmented features of compressed videos as follows:

$$\mathcal{L}_{unsup} = \frac{1}{3N_{unsup}} \sum_{j=1}^{N_{unsup}} \sum_{V \in \{I,M,R\}} H(p_{\theta^V}(y|\mathcal{A}(u_j^V)), \hat{q}_j^V), \tag{3.14}$$

where $\mathcal{A}$ is the strong augmentation described in Section 3.5.1.

Now, we can define our total loss $\mathcal{L}$ as follows:

$$\mathcal{L} = \mathcal{L}_{sup} + \lambda \mathcal{L}_{unsup}, \tag{3.15}$$

where λ is a balancing weight for the unlabeled data. Our model is optimized to minimize the loss $\mathcal{L}$ using SGD. Finally, we summarize a complete algorithm for CoVEnPL in Algorithm 3.

3.5.3 Experimental Setup

Following previous studies [Wu et al., 2018, Shou et al., 2019], we used a larger network for an I-frame network and smaller networks for the motion vector and residual networks. Table 3.4 shows the architectures of our models. In our experiments, we used two models based on ResNet-18 and ResNet-50. We trained our models using Adam [Kingma and Ba, 2014] with $\alpha = 0.002$ and a weight decay of 0.0004. In addition, we set $\tau = 0.5$, $\lambda = 1$ for our loss, and $B_s = B_u = 32$ for the batch size. We used PyTorch [Paszke et al., 2019] to implement CoVEnPL and mixed precision training [Micikevicius et al., 2017] to reduce GPU memory usage.

Augmentation

We used videos resized to a pixel resolution of 128 on the shorter side while maintaining the aspect ratio. We set the size of each GOP as 12, which means that each GOP contains an I-frame and 11 P-frames. For temporal augmentation, we set $T = 5$ and $\mu = 5$ to select frames, and thus we input 5 I-frames and 25 P-frames into our networks. For a weak augmentation, we randomly cropped patches with a pixel resolution of 112×112 from the extracted frames and applied horizontal flipping to the patches with 50% probability. For a strong augmentation, we applied RandAugment described in Section 3.5.1 to compressed video features, in addition to the weak augmentation. However, motion vectors only have two channels, and some transformations in RandAugment cannot transform motion vectors because their in-

Algorithm 3 CoVEnPL algorithm.

Require: Network parameters $\{\theta^I, \theta^M, \theta^R\}$, labeled data $\mathcal{X}$, unlabeled data $\mathcal{U}$, sharpening temperature τ, balancing weight λ, learning rate η.

Ensure: Updated parameters $\{\theta^I, \theta^M, \theta^R\}$

while $\{\theta^I, \theta^M, \theta^R\}$ do not converge **do**
  // Compute supervised loss
  $B_{sup} = \{(x_i, y_i) | 1 \leq i \leq N_{sup}\} \sim \mathcal{X}$
  Get $\{(x_i^I, x_i^M, x_i^R) | 1 \leq i \leq N_{sup}\}$ from B_{sup}
  $\mathcal{L}_{sup} = \frac{1}{3N_{sup}} \sum_{(x_i,t_i)\in B_{sup}} \sum_{V\in\{I,M,R\}} H(p_{\theta^V}(y_i^V | a(x_i^V)), t_i)$
  // Compute unsupervised loss
  $B_{unsup} = \{u_j | 1 \leq j \leq N_{unsup}\} \sim \mathcal{U}$
  Get $\{(u_i^I, u_i^M, u_i^R) | 1 \leq j \leq N_{unsup}\}$ from B_{unsup}
  for $j = 1$ **to** N_{unsup} **do**
    $q_j = \frac{1}{3} \sum_{V\in\{I,M,R\}} p_{\theta^V}(y | a(u_j^V))$ ▷ Ensemble
    $\hat{q}_j = \frac{q_j^{1/\tau}}{\sum_{k=1}^{C} q_{jk}^{1/\tau}}$ ▷ Sharpen function
  end for
  $\mathcal{L}_{unsup} = \frac{1}{3N_{unsup}} \sum_{j=1}^{N_{unsup}} \sum_{V\in\{I,M,R\}} H(p_{\theta^V}(y | \mathcal{A}(u_j^V)), \hat{q}_j^V)$
  // Compute loss to minimize
  $\mathcal{L} = \mathcal{L}_{sup} + \lambda \mathcal{L}_{unsup}$
  // Update parameters by SGD
  $\theta^I \leftarrow \theta^I - \eta \frac{\partial \mathcal{L}}{\partial \theta^I}$
  $\theta^M \leftarrow \theta^M - \eta \frac{\partial \mathcal{L}}{\partial \theta^M}$
  $\theta^R \leftarrow \theta^R - \eta \frac{\partial \mathcal{L}}{\partial \theta^R}$
end while

Table 3.4 The architecture of our networks. We follow the previous studies and use a large network for I-frames and smaller networks for motion vectors and residuals. In our model, the spatial size of the inputs is halved by Conv1, Res2, Res3, and Res4. By contrast, the temporal size of the inputs is maintained. In this table, f^I, f^M, and f^R denote an I-frame, motion vector, and residual network.

Layer Name	ResNet-18 based model		ResNet-50 based model	
	f^I	f^M and f^R	f^I	f^M and f^R
Conv1	$1 \times 7^2, 64$ stride $1, 2^2$	$5 \times 7^2, 64$ stride $2, 2^2$	$1 \times 7^2, 64$ stride $1, 2^2$	$5 \times 7^2, 64$ stride $2, 2^2$
Res1	$\begin{bmatrix} 1 \times 3^2, 64 \\ 3 \times 3^2, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3^2, 16 \\ 3 \times 3^2, 16 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 16 \\ 1 \times 3^2, 16 \\ 1 \times 1^2, 64 \end{bmatrix} \times 3$
Res2	$\begin{bmatrix} 1 \times 3^2, 128 \\ 3 \times 3^2, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3^2, 32 \\ 3 \times 3^2, 32 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 1^2, 32 \\ 1 \times 3^2, 32 \\ 1 \times 1^2, 128 \end{bmatrix} \times 4$
Res3	$\begin{bmatrix} 3 \times 3^2, 256 \\ 3 \times 3^2, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3^2, 64 \\ 3 \times 3^2, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 1^2, 256 \\ 1 \times 3^2, 256 \\ 1 \times 1^2, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 6$
Res4	$\begin{bmatrix} 3 \times 3^2, 512 \\ 3 \times 3^2, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3^2, 128 \\ 3 \times 3^2, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 1^2, 512 \\ 1 \times 3^2, 512 \\ 1 \times 1^2, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{bmatrix} \times 3$
Output	Global Average Pooling, FC			

Table 3.5 Accuracy comparison of RGB-based methods on Kinetics-100. We report the average and standard deviation of the top-1 accuracy of CoVEnPL and compare it with the results reported by other works. In this table, CMPL and Xiao et al.'s method are omitted because they do not report their results on Kinetics-100.

Method	Kinetics-100			
	5%	10%	20%	50%
Pseudo Label [Lee et al., 2013]	27.8	38.9	48.0	59.0
Mean Teacher [Tarvainen and Valpola, 2017]	27.8	36.4	47.1	59.3
S4L [Zhai et al., 2019]	33.0	43.3	51.1	54.6
VideoSSL [Jing et al., 2021]	**47.6**	52.6	**57.7**	**65.0**
Zou et al. [Zou et al., 2021]	—	**61.2**	—	—
CoVEnPL	37.5 ± 5.2	43.27 ± 5.7	46.9 ± 4.7	52.2 ± 3.6

put must be three channels, as in RGB images. Therefore, to apply RandAugment to motion vectors, following a previous study [Xiong et al., 2021], we add an extra channel representing the magnitude of movements to the motion vectors.

3.5.4 Accuracy Comparison with RGB-based Methods

Results on UCF-101

For UCF-101, CoVEnPL consistently outperformed the baseline methods that use only RGB frames as an input. For example, CoVEnPL achieved 54.8% accuracy on only 5% of the labels. It scored 24.5 points higher than VideoSSL on the same number of labels. Surprisingly, our method trained using only 5% of the labels also outperformed the methods that use only RGB frames as an input, except for Zou et al.'s method trained using 50% of the labels of the dataset. In addition, the gap between CoVEnPL trained using 5% of the labels and Zou et al.'s method trained using 50% of the labels is only 3.0 points. Comparing CoVEnPL with CMPL and Xiao et al.'s method, their performances on UCF-101 were competitive. CMPL and

Xiao et al.'s method uses multiple types of inputs. In contrast, others use only RGB frames; these results show that using multiple types of features in compressed videos can improve performance effectively.

Results on HMDB-51

Like UCF-101, CoVEnPL also performed better against baseline methods that use only RGB frames as input with fewer labels in HMDB-51. For example, CoVEnPL achieved 31.0% accuracy on 10% of the labels. This result is better than Pseudo Label and Mean Teacher trained using 50% of the labels. Moreover, CoVEnPL trained on 20% of the labels achieved higher performances than these single-stream baseline methods trained on 50%. CoVEnPL is also competitive against Xiao et al.'s method using multiple types of inputs, although the performance gap is not so significant. These results show that, for HMDB-51, CoVEnPL is more effective than the baseline methods that use only RGB frames and competitive against the baseline methods that use multiple features.

Results on Kinetics-100

For Kinetics-100, CoVEnPL mostly outperformed Pseudo Label, Mean Teacher, and S4L. However, when training our model using 50% of the labels, CoVEnPL achieved lower performances than Pseudo Label and Mean Teacher. CoVEnPL also consistently underperformed VideoSSL and Zou et al.'s method. These results show that, unlike the experiments conducted on UCF-101 and HMDB-51, CoVEnPL could not improve the performances on Kinetics-100 from the RGB-based methods.

3.5.5 Accuracy Comparison with Multi-stream Methods

We also compared CoVEnPL with MvPL, which uses RGB frames, an optical flow, and temporal gradients to generate artificial labels and CMPL on the UCF-

Table 3.6 Accuracy comparison of RGB-based methods on UCF-101 and HMDB-51. We report the average and standard deviation of the top-1 accuracy of CoVEnPL and compare it with the results reported by other works. We cannot report the standard deviation of the baseline methods' scores because the values are not reported.

Method	UCF-101				HMDB-51			
	5%	10%	20%	50%	5%	10%	20%	50%
Pseudo Label [Lee et al., 2013]	17.6	24.7	37.0	47.5	–	–	–	27.3
Mean Teacher [Tarvainen and Valpola, 2017]	17.5	25.6	36.3	45.8	–	–	–	27.2
S4L [Zhai et al., 2019]	22.7	29.1	37.7	47.9	–	–	–	31.0
VideoSSL [Jing et al., 2021]	32.4	42.0	48.7	54.3	–	–	–	35.2
Zou et al. [Zou et al., 2021]	20.7	40.2	51.7	59.9	–	–	–	38.2
CMPL [Xu et al., 2021]	–	67.6	–	–	–	–	–	–
Xiao et al. [Xiao et al., 2021]	44.8	62.4	**76.1**	**79.3**	–	–	–	48.4
CoVEnPL	**54.8 ± 2.1**	**68.0 ± 1.8**	75.8 ± 1.9	78.9 ± 3.8	**24.5 ± 0.3**	**31.0 ± 0.7**	**38.9 ± 0.6**	**49.2 ± 1.1**

Table 3.7 Accuracy comparison with MvPL and CMPL on UCF-101 using ResNet-50-based models. We report the top-1 accuracy of CoVEnPL and compare it with the reported results. Compared with the RGB-based methods, CoVEnPL cannot outperform MvPL and CMPL.

Method	1%	10%
MvPL [Xiong et al., 2021]	<u>22.8</u>	**80.5**
CMPL [Xu et al., 2021]	**25.1**	<u>79.1</u>
CoVEnPL	21.3 ± 0.9	63.6 ± 3.1

101 dataset. Because MvPL only reports the results of the ResNet-50-based model, we also use the ResNet-50-based model for a fair comparison. Our model has 40M parameters, roughly 6M more than the MvPL model, because it consists of three networks to process our inputs while MvPL uses a network to process their inputs. Unlike the previous experiments, we used 1% and 10% of the labels of UCF-101 to train our model and ran three trials.

Table 3.7 shows the results of our comparison. This table indicates that CoVEnPL achieved competitive performances against MvPL when training the models using only 1% of the labels. When training the models using 10% of the labels, CoVEnPL achieved only an accuracy of 63.6%, which is 15.1 points lower than that of MvPL and 13.7 points lower than that of CMPL. From these results, MvPL and CMPL were superior to CoVEnPL. Comparing this result with Table. 3.6, the classification performances of CoVEnPL on UCF-101 with 10% labels decreased when changing models from ResNet-18 to ResNet-50. From this observation, CoVEnPL cannot outperform CMPL in the previous experiments because CMPL improved classification scores by increasing the capacity of models while CoVEnPL failed to improve scores.

Table 3.8 Speed comparison with baseline methods. We show the preprocessing and feed-forwarding times (ms) per frame. Because MvPL uses RGB frames, an optical flow, and temporal gradients for training and uses only RGB frames for inference, we distinguish between MvPL training and MvPL inference. From this result, we observe that CoVEnPL is faster than the other methods owing to the use of compressed videos.

Method	RGB-based method	MvPL inference	MvPL training	CoVEnPL
Preprocess (ms)	25.2	25.6	73.5	3.2
Feed-forward (ms)	0.4	1.1	1.4	0.5
Total (ms)	25.6	26.7	74.9	3.7

3.5.6 Speed Comparison

As an advantage of CoVEnPL, we can efficiently prepare the inputs because the features of the compressed video can be loaded from the compressed video files without any decoding. Table 3.8 shows the per-frame runtime speed (ms) on an Nvidia Tesla V100 GPU with Intel Xeon Gold 6230 CPUs. CoVEnPL showed the shortest preprocessing time. In particular, in terms of the preprocessing time, our approach is roughly 25 times faster than MvPL training. Although MvPL inference is faster than training because MvPL does not use an optical flow for inference, it is not faster than our approach. In addition, there are only slight differences within all methods in the feed-forward time despite our model consisting of three networks. This is because our networks that process motion vectors and residuals are smaller than our network that processes I-frames.

3.5.7 Ablation Study

We conducted ablation studies to analyze CoVEnPL. For the ablation studies, we trained the ResNet-18 based model using UCF-101 with 10% of the labels, and the result of our full method shows a 68.0% accuracy, as indicated in Table 3.6. We ran

five trials per experiment and summarized our ablation studies in Table 3.9. From these studies, we found the following results.

First, we removed our unsupervised loss (i.e., set λ=0) to observe whether CoVEnPL utilizes unlabeled videos well. This is one of the most important factors of SSL methods. Consequently, CoVEnPL without an unsupervised loss achieved only 47.1% accuracy, which is 20.9 points lower than our full method. This result shows that CoVEnPL can utilize unlabeled videos effectively to improve performance.

Second, we removed the ensemble part from Eq. 3.12. We trained our networks independently and ensembled their predictions only for the final inference. Therefore, CoVEnPL without an ensemble achieved an accuracy of 62.1%, which is 5.9 points lower than our full method. This result shows that using ensembled artificial labels is also effective for compressed videos.

Third, we replaced strong augmentation with weak augmentation to show how much strong augmentation contributes to CoVEnPL. Consequently, CoVEnPL, using only weak augmentation, achieved only a 51.9% accuracy level, which is 16.1 points lower than our full method. The gap in performance between our full method and this ablation study is the widest in our ablation studies except for removing unsupervised loss. Because removing unsupervised loss also removes other components, this result shows that strong augmentation is the most important component for the performance of CoVEnPL. We did not specifically tune it for CoVEnPL; thus, we can improve its performance of CoVEnPL by exploring a more effective and stronger augmentation for compressed videos. The strong augmentation also improves the performance of supervised image and video classification. Hence, exploring such strong augmentation for compressed videos is promising for semi-supervised and supervised CoViAR.

Finally, we removed the sharpen function and used $\hat{q_j} = q_j$ instead of Eq. 3.13. CoVEnPL without the sharpen function achieved a 66.3% accuracy, which is 1.7 points lower than our full method. This result showed that although the contribution

Table 3.9 Ablation study. We conducted each experiment on UCF-101 with 10% of the labels. All components of CoVEnPL contributed to the improved performance.

Method	Accuracy
Full method	68.0 ± 1.8
w/o unsupervised loss	47.1 ± 2.6
w/o ensemble predictions	62.1 ± 4.3
w/o strong augmentation	51.9 ± 3.8
w/o sharpen function	66.3 ± 2.3

of the sharpen function is limited compared with other components, it also contributes to the performance of CoVEnPL.

3.5.8 Discussion

In this study, we proposed CoVEnPL to reduce large computational and storage costs for training required by the conventional semi-supervised video classification methods. Unlike the previous works, such as MvPL, we used compressed video features as inputs. We could reduce the computational cost as shown in Table 3.8 without storing all frames of datasets as images in advance. In addition, for UCF-101 and HMDB-51, CoVEnPL achieved competitive performances against the baseline methods.

The performance comparison on UCF-101 and HMDB-51 indicates that compressed video features can be used to obtain accurate models in an SSL manner, and CoVEnPL utilizes them effectively. In Table 3.6, CoVEnPL, CMPL, and Xiao et al.'s method achieved the competitive classification performances. The common point of these methods is that they use multiple types of inputs to train models. When we trained ResNet-50 based models as shown in Table 3.5, MvPL, which

also used multiple types of inputs, achieved better performances than CoVEnPL. In this experiment, CMPL achieved competitive performances against MvPL and outperformed CoVEnPL. However, in practice, there is an advantage to using CoVEnPL against others, including MvPL and CMPL; CoVEnPL requires fewer times to process videos than not only MvPL that uses optical flow but also others that do not use optical flow like CMPL and Xiao et al.'s method. This advantage comes from the compressed video features, making training easier to scale up than other methods.

Our CoVEnPL achieves a better level than the performance of conventional state-of-the-art methods due to the strong augmentation and the ensemble of predictions. Our method's key component is its ensemble of predictions to generate artificial labels. Our ablation study showed that the ensemble improved performances. This is because the compressed video features used in our proposed method consist of I-frames, motion vectors, and residuals, each of which holds important action information required for video classification. Our ablation study also indicates that the CoVEnPL performances heavily rely on strong augmentation. Strong augmentation is a popular consistency regularization in semi-supervised image classification [Zhai et al., 2019, Berthelot et al., 2020, Sohn et al., 2020, Zhang et al., 2021]. The strong augmentation is applied to I-frames, motion vectors, and residuals individually. We discuss how strong augmentation works for each feature.

The strong augmentation adds large perturbations to compressed video features and improves the robustness of our models. In this paper, we use the same strong augmentation that has been conventionally used in the semi-supervised image classification methods [Zhai et al., 2019, Berthelot et al., 2020, Sohn et al., 2020, Zhang et al., 2021] because there are no studies where strong augmentations have been applied to compressed video action recognition. It is not clear how the strong augmentations should be implemented. Our method's strong augmentation

consists of color and geometric transformations. Our ablation study shows that both transformations can improve the performance of CoVEnPL. However, because the augmentation has been tuned only for RGB images without considering temporal features, it could distort the temporal action features stored in compressed videos excessively and may lead to poor performances. Nevertheless, our ablation study indicates that such strong augmentation is even more beneficial than weak augmentation, which does not distort the temporal action features. We must consider what each compressed feature represents to understand why the strong augmentation works well. As I-frames are just a sequence of RGB images, the strong augmentation naturally works well for I-frames, as in previous studies. Regarding residuals, their important action features are the edges of moving objects in videos. The strong augmentation does not remove the features of these edges, and our network could classify videos from the strongly augmented residuals. Therefore, the strong augmentation helps our residual networks to obtain robustness against large perturbations.

Unlike I-frames and residuals, there is a possibility that the strong augmentation is not suitable for motion vectors. In our method, the motion vectors represent the amounts and directions of pixel movements in videos as color images. Therefore, color transformations change the amounts and directions of movements, which are crucial temporal action features indicating how objects move in video frames. Even though the color transformation distorts the temporal features, the strongly transformed motion vectors are still classifiable in many cases. Sevilla et al. focused on what optical flow features lead to better video classification [Sevilla-Lara et al., 2018]. In their experiments, the temporal coherence in the training videos was removed by shuffling the orders of video frames and making optical flow inputs from the randomized frames. Such optical flow highlights the pixels for which motion is detected. In other words, information shapes the edges of moving objects in the image frames. Even when the

temporal coherence was removed from the optical flow, their models could still learn to classify videos. Their findings indicate that actions of optical flow are classifiable from the movement represented as temporal features appearing in the consecutive frames and the shape of moving objects represented by information where the movement occurred. Such information is not removed by the strong augmentation. There is also the case where color transformations are not applied, as the strong augmentations used once are randomly sampled from a set of possible transformations. In that case, the temporal features are not distorted, and the augmentations benefit training.

For the above reasons, the strong augmentation can add large but suitable perturbations to I-frames, residuals, and even motion vectors and helps models improve their performances. However, the strong augmentation here was applied using a simple conventional method. Developing better strong augmentations for residuals and motion vectors is an open problem for our method and deep learning models that use compressed videos as input.

The entropy minimization is another popular regularization in SSL [Lee et al., 2013, Berthelot et al., 2019, Xie et al., 2020, Zhai et al., 2019, Berthelot et al., 2020, Sohn et al., 2020, Zhang et al., 2021]. Still, the performance improvement of this regularization was relatively limited compared with strong augmentation in our method. To minimize the entropy of predictions, models should output high-confidence predictions. Our method applies the sharpen function to the artificial labels for entropy minimization. Unlike popular inputs such as RGB frames and optical flow, compressed video features are designed to remove redundant information as much as possible to reduce the file size. Hence, action features contained in each input will be relatively limited compared with RGB frames and optical flow, and it is more difficult to classify videos from each input feature with high confidence. The ensemble of predictions for generating the artificial labels also weakens the contribution of the entropy minimization. Although the ensemble will improve the

correctness of artificial labels, it tends to generate uniform distribution-like artificial labels when models classify the same videos into different actions. Such uniform distribution-like artificial labels train models to output lower-confidence predictions, so the work of the sharpen function would be limited.

Another limitation of CoVEnPL was shown when we trained models using Kinetics-100. Unlike UCF-101 and HMDB-51, CoVEnPL was worse than other baseline methods using RGB frames. Compared to the other datasets, Kinetics-100 seems to require the extraction of detailed temporal features. If we can solve this problem, our method is expected to improve performance and become more practical.

3.6 Conclusion

In this chapter, we proposed CoVEnPL, a novel SSL method for video classification. Inspired by the recent SSL method MvPL, we introduced an ensembling approach to generate reliable artificial labels for unlabeled data. At the same time, CoVEnPL uses features stored in compressed videos as inputs to reduce the computational and storage costs for training. We found that the ensembling approach effectively improves performance using compressed videos. In addition, by using compressed videos directly as inputs, CoVEnPL can reduce the computational and storage costs for training and easily scale up the training.

In our experiments, CoVEnPL cannot reach the classification performance of MvPL but achieves competitive performances against other state-of-the-art optical flow-free SSL methods on the public datasets UCF-101 and HMDB-51. Furthermore, we show that the loading of our input features is faster than that of the other method. However, interestingly, CoVEnPL does not work well on Kinetics-100.

We believe that improving the SSL methods using compressed videos is a promising approach because the number of videos available online is increasing now, and it is easier to utilize unlabeled videos for training.

Chapter 4

Efficient Compressed Video Action Recognition with a Single Network

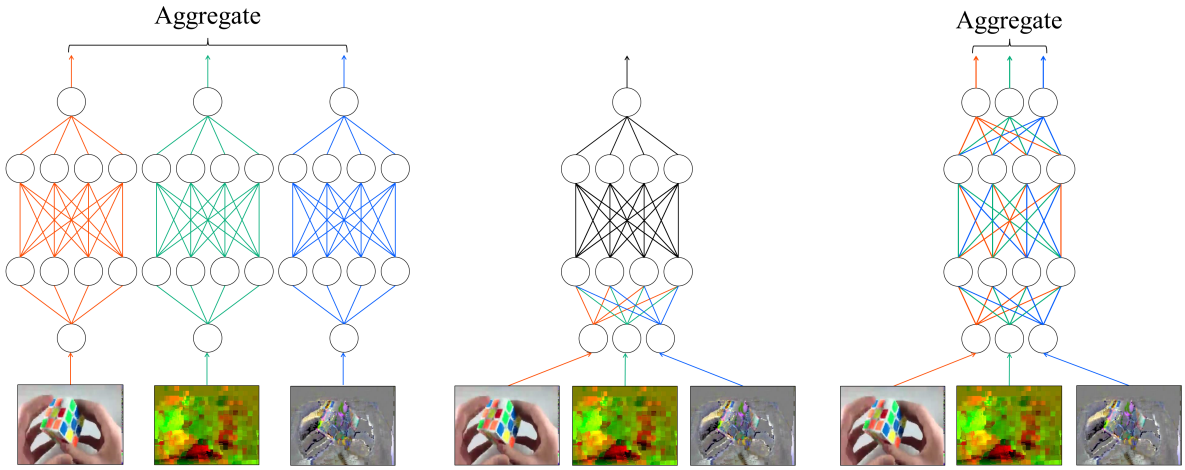
4.1 Introduction

Compressed video action recognition is more efficient than RGB frame-based video classification in terms of computational complexity. Some studies have focused on the low computational complexity of compressed video action recognition to deploy networks on mobile or edge devices [He et al., 2021, Huo et al., 2019].

The compressed video features consist of multiple features whose information should be fused effectively to improve video classification performance. Most conventional methods use multiple networks to process the compressed video features [Wu et al., 2018, Huo et al., 2019, dos Santos et al., 2019, Shou et al., 2019, Wu et al., 2019]. This approach is depicted in Figure 4.1-(a) and is called late fusion. Recent studies have improved late fusion by fusing internal states to establish additional information paths between the networks [He et al., 2021, Li et al., 2020]. The disadvantage of late fusion is that it needs multiple networks to process compressed video features. This problem prevents us from reducing the computational

complexity for efficient compressed video action recognition. Early fusion, shown in Figure 4.1-(b), does not suffer from this problem. This approach concatenates compressed video features along channel dimensions in advance and simultaneously processes the concatenated features using a single network. This approach does not need multiple networks, providing scope to reduce the computational complexity compared to late fusion using the same backbone networks. However, the video classification performance of early fusion is much poorer than that of late fusion.

To overcome this computational complexity and performance problem, we propose to train a single network for late fusion. Our method is inspired by the recently proposed approach named multi-input multi-output (MIMO) [Havasi et al., 2021]. MIMO was initially proposed as an efficient ensemble method. This method feeds multiple "different" images into a network and trains subnetworks embedded in the single network to classify the images independently. As a result, MIMO can simultaneously compute multiple subnetworks in one single network, allowing ensemble results to be output. In this work, we extend MIMO to compressed video action recognition, as depicted in Figure 4.1-(c). Instead of applying MIMO by using multiple entire videos as the inputs, we extract different compressed video features from multiple videos and use them as inputs during the training phase. After training, our network can perform the late fusion of different compressed video features with a single feed-forwarding using the subnetworks. Our experiments show that our method attains the same level of computational complexity as early fusion and accuracy as late fusion. In addition, the video classification performance of our method is competitive with state-of-the-art compressed video action recognition methods on popular video classification datasets.



(a) Late fusion (b) Early fusion (c) Proposed method

Fig. 4.1 Typical approaches to fuse compressed video features (a, b) and our proposed method (c). Colors indicate networks or subnetworks that process each input feature. Our method uses a single network similar to early fusion but internally holds independent subnetworks for late fusion.

4.2 Background

4.2.1 Typical Fusion Methods for Multi-stream Inputs

Fusing the information to utilize multiple types of inputs, including compressed videos, is crucial to obtain accurate predictions effectively. The design of the fusion method has been extensively studied as it significantly affects classification performance [Jiang et al., 2018, Boulahia et al., 2021, Joze et al., 2020, Duong et al., 2017, Hu et al., 2019]. This study focused on three naïve fusion methods: early, late, and intermediate fusion.

Late Fusion

Late fusion, depicted in Fig. 4.1-(b), is another simple fusion method for compressed videos, which is employed by many conventional methods [Wu et al., 2018,

Li et al., 2021, dos Santos et al., 2019, Shou et al., 2019, Guo et al., 2023, Huo et al., 2019]. In compressed video action recognition, the late fusion method used three networks and independently classified actions from each compressed video feature using these networks. The final prediction was obtained by averaging the three predictions. Late fusion can significantly improve classification performance compared to early fusion. However, late fusion only linearly fuses the predictions from compressed videos and cannot nonlinearly fuse compressed video information, leaving room for further improvement in accuracy.

Early Fusion

For the implementation level, our method is based on early fusion. Let $\mathcal{X} = \{x_i, t_i | 0 \leq i \leq N\}$ be a video dataset, where x_i is an i-th video, t_i is an action category of x_i and N is the number of videos in the dataset. From each video x_i, we can obtain compressed video features: I-frames x_i^I, motion vectors x_i^M, and residuals x_i^R. The early fusion concatenates compressed video features along their channel dimensions and processes the concatenated features by a single network. Here, let $p_\theta(y|x_i^I, x_i^M, x_i^R)$ be the prediction of the single network from x_i^I, x_i^M, and x_i^R, where θ is the network parameters. We optimize θ to classify videos correctly. Formally, early fusion minimizes the loss $\mathcal{L}_{early}$ defined as:

$$(x_i^I, x_i^M, x_i^R) \sim \mathcal{X}, \tag{4.1}$$

$$\mathcal{L}_{early} = H(p_\theta(y|x_i^I, x_i^M, x_i^R), t_i), \tag{4.2}$$

where H is the cross-entropy loss.

Although this method does not need as much computational complexity as the late fusion, we observe that this method does not achieve competitive video classification performance against the late fusion. We tackle this performance problem while keeping the low computational complexity as early fusion.

4.2.2 Efficient Ensemble of Deep Networks

Our method is inspired by the recent ensemble method that can estimate the uncertainty of predictions or improve out-of-distribution robustness by feeding the same inputs (e.g., images) into multiple networks and fusing their predictions. The problem with the ensemble methods is the expensive computation and memory costs for training and testing multiple networks. Various approaches, such as Monte Carlo Dropout [Gal and Ghahramani, 2016] and Snapshot [Huang et al., 2017], were proposed to address this problem. Havasi et al. proposed the MIMO method [Havasi et al., 2021], which uses a single MIMO network instead of multiple single-input single-output networks. Unlike other methods, their method processes multiple inputs with one feed-forwarding of a single network. They showed that independent subnetworks are obtained in a single network through MIMO learning. Sun et al. integrated interpolation-based data augmentation such as mixup [Zhang et al., 2017] and cutmix [Yun et al., 2019] into the MIMO method and improved performance [Ramé et al., 2021, Sun et al., 2022]. Ferianc and Rodrigues used more output layers to improve the MIMO model [Ferianc and Rodrigues, 2023]. DataMUX [Murahari et al., 2022] employed a MIMO-like approach to process long texts simultaneously using transformers [Vaswani et al., 2017]. We extend the MIMO method into compressed video action recognition to obtain a single network that processes compressed video features simultaneously.

4.3 Late Fusion Approximation by a Single Network

4.3.1 Multi-stream Single Network

We train a single network for late fusion. As depicted in Fig. 4.1-(c), our network comprises the shared backbone network and three single-layer networks. The shared backbone network computes the embeddings from I-frames, motion vectors,

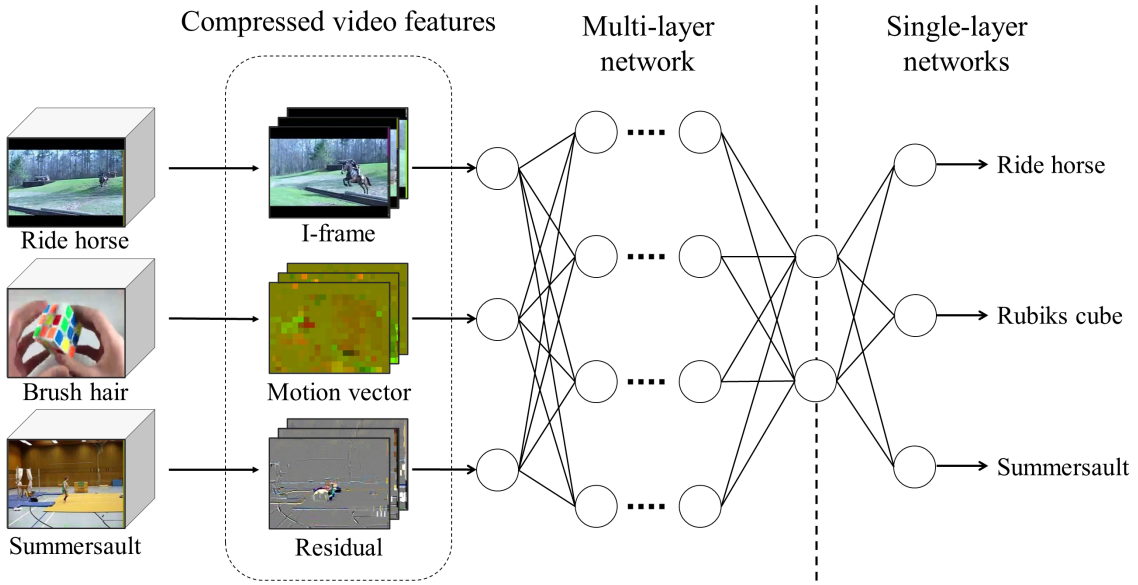


Fig. 4.2 Training phase of our proposed method. We feed compressed video features extracted from different videos into a single network simultaneously and train the network to classify each video.

and residuals. The single-layer networks correspond to one of the I-frames, motion vectors, and residuals and predict classes from the corresponding features. The computational complexity of this network is almost maintained compared to early fusion, as the shared backbone network computes the embeddings only once.

To train single-layer networks that classify videos independently from the shared embeddings, we extend the MIMO approach into compressed video action recognition, as shown in Figure 4.2. During training, we sample compressed video features x_i^I, x_j^M, and x_k^R from different videos x_i, x_j, and x_k, respectively, and concatenate them to feed into a single network. The network outputs three predictions $p_\theta(y_i^I|x_i^I, x_j^M, x_k^R)$, $p_\theta(y_i^M|x_i^I, x_j^M, x_k^R)$, and $p_\theta(y_i^R|x_i^I, x_j^M, x_k^R)$ obtained from the concatenated input, and each prediction is optimized to predict the classes of different corresponding videos t_i, t_j, and t_k.

Because video classes of x_i, x_j, and x_k may differ, each video's compressed video

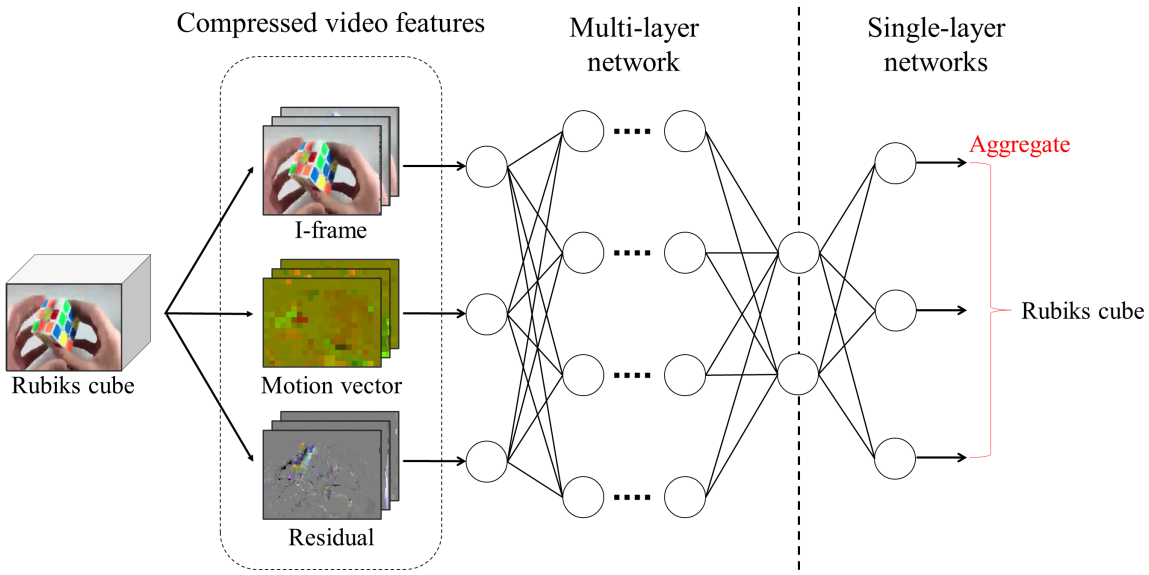


Fig. 4.3 Evaluation phase of our proposed method.

feature does not have helpful information to classify the other two videos. Thus, our network will make subnetworks that compute features independently to classify all videos correctly. After training, the obtained subnetworks can be used for late fusion instead of multiple networks depicted in Figure 4.1-(a). Formally, we optimize networks to minimize the loss value $\mathcal{L}$ defined as:

$$\mathcal{L} = \sum_{(l,V)\in\{(i,I),(j,M),(k,R)\}} H(p_\theta(y_i^V|x_i^I, x_j^M, x_k^R), t_l). \tag{4.3}$$

Note that the above method for sampling different features from different videos is not performed during the evaluation phase as depicted in Fig 4.3. We use compressed video features extracted from the same videos as inputs and average the three predictions corresponding to I-frames, motion vectors, and residuals to obtain the final prediction $p_\theta(y_i|x_i^I, x_i^M, x_i^R)$ as follows:

$$p_\theta(y_i|x_i^I, x_i^M, x_i^R) = \frac{1}{3} \sum_{V\in\{I,M,R\}} p_\theta(y_i^V|x_i^I, x_i^M, x_i^R). \tag{4.4}$$

4.3.2 MIMO Pre-training

Following previous studies [Wu et al., 2018, Huo et al., 2019], we finetune networks pre-trained by the ImageNet1K dataset [Russakovsky et al., 2015]. The standard classification training is unsuitable for our method's pre-training because it does not encourage networks to make multiple subnetworks. Instead of the standard classification training, we use MIMO training as the pre-training to obtain subnetworks in the pre-training phase. We experimentally show that our method requires this pre-training strategy to reach the late fusion level accuracy.

4.3.3 Experimental Setup

We used the UCF101 [Soomro et al., 2012] and HMDB51 [Kuehne et al., 2011] datasets to evaluate our method. All videos were resized to 240×320 resolution and compressed to the MPEG4 Part-2 format [Le Gall, 1991]. We randomly sampled 8 I-frames, motion vectors, and residuals from videos allocated for training. The frames were cropped into 224×224 patches and underwent horizontal flipping with 50% probability. For evaluation, we uniformly sampled 8 I-frames, motion vectors, and residuals from videos and cropped the central 224×224 patches from the frames.

As the backbones of our networks, we chose ResNet18, ResNet34, and ResNet50 [He et al., 2016] with temporal shift modules (TSMs) [Lin et al., 2019]. TSMs can adapt 2D convolutional networks for video processing without increasing the number of parameters and FLOPs. We first trained the backbone networks without TSMs using the ImageNet1K dataset [Russakovsky et al., 2015]; subsequently, we finetuned the network parameters to classify the UCF101 or HMDB51 datasets with the TSMs. To demonstrate the effectiveness of MIMO pre-training, we finetuned both the MIMO pre-trained parameters and standard pre-trained parameters provided by the PyTorch team. Our networks were optimized by stochastic gradient descent

with Nesterov momentum (learning rate: 0.1; momentum: 0.9; weight decay rate: 1.0×10^{-5}). We trained our networks for 150 epochs. A factor of 0.2 decayed the learning rate after 60, 90, and 120 epochs.

As the baseline methods, we used the standard early fusion described in Section **??** and late fusion depicted in Figure 4.1-(a). Baseline methods used the same backbones as those used in ours. We trained the baseline methods with approximately the same optimization strategies described above. However, because the training loss of baseline methods on HMDB51 did not converge after 150 epochs, we trained our baseline methods four times longer than our method; that is, we trained the baseline networks for 600 epochs. Accordingly, the learning rate decreased after 240, 360, and 480 epochs.

4.3.4 Comparison with Typical Fusion Methods

We compared our method with the baseline methods and summarized the results in Table 4.1.

The early fusion baseline exhibited much poorer video classification performances than the late fusion baseline, regardless of the backbone networks. These results indicated that early fusion was impractical for compressed video action recognition. Although our method used only a single network like early fusion, it achieved competitive video classification performances compared to the late fusion baseline.

Our method required approximately three times lower computational complexity than the late fusion baseline. This is because the late fusion used three networks. As a result, our method with ResNet50 still yielded a lower computational complexity than the late fusion baseline with ResNet18.

The performance improvement of our method depended on the pre-training method. When we did not pre-train parameters for image classification using MIMO, the video classification performance of our method was consistently poorer than that of the late

Table 4.1 Comparison between the proposed method and baselines for computational complexity (GFLOPs), prediction time, number of parameters, and video classification performance. We report the results of the split 1 (S1), split 2 (S2), split 3 (S3), and average (Avg.) scores of all splits.

Backbone	Method	GFLOPs	Prediction time (ms)	Params (M)	UCF101				HMDB51			
					S1	S2	S3	Avg.	S1	S2	S3	Avg.
ResNet18	Early fusion	15.3	2.63	11.3	74.5	74.6	76.0	75.1	42.3	39.9	39.4	40.5
	Late fusion	40.5	7.19	33.7	87.8	88.6	88.7	88.4	61.4	57.7	57.3	58.8
	Proposed w/o MIMO pre-training	15.4	2.64	11.4	82.2	82.1	84.1	82.8	47.7	46.2	43.3	45.7
	Proposed	15.4	2.64	11.4	85.7	85.8	85.8	85.8	59.0	57.9	58.9	58.6
ResNet34	Early fusion	28.9	4.42	21.4	77.3	78.7	77.7	77.9	41.1	42.5	41.1	41.6
	Late fusion	81.2	12.38	64.1	90.1	90.0	89.7	89.9	62.5	62.2	62.2	62.3
	Proposed w/o MIMO pre-training	28.9	4.42	21.5	83.7	83.1	84.9	83.9	42.1	41.9	44.9	43.0
	Proposed	28.9	4.42	21.5	88.4	88.6	89.6	88.9	60.9	61.6	62.6	61.7
ResNet50	Early fusion	33.6	5.86	23.8	81.6	81.1	81.8	81.5	42.4	46.9	45.4	44.9
	Late fusion	95.3	16.60	71.3	87.7	87.7	88.7	88.0	61.9	63.2	63.0	62.7
	Proposed w/o MIMO pre-training	33.6	5.86	24.2	87.1	89.9	88.5	88.2	47.1	45.5	46.0	46.2
	Proposed	33.6	5.86	24.2	88.7	90.0	89.4	89.2	62.5	62.8	63.5	62.9

Table 4.2 Comparison between the proposed method and conventional compressed video action recognition methods in terms of video classification performance and computational complexity (GFLOPs). Views indicate the number of test time augmentations to achieve the reported accuracy scores. For example, CoViAR, Wu et al., and TTP applied the ten-crop to every frame.

Method	Backbone	Pretrain	GFLOPs × Views	UCF-101	HMDB-51
CoViAR [Wu et al., 2018]	ResNet18+ResNet152	ImageNet1K	361.5× 10	90.4	59.1
DMC-Net [Shou et al., 2019]	I3D	Kinetics400	401 × 1	92.3	71.8
CV-C3D [dos Santos et al., 2019]	C3D	ImageNet1K	96.4 × 1	83.9	55.7
Wu et al. [Wu et al., 2019]	ResNet18	ImageNet1K	125.4× 10	88.5	56.2
TTP [Huo et al., 2019]	MobileNetV2	ImageNet1K	105 × 10	87.2	58.2
SIFP [Li et al., 2020]	SIFP	Kinetics400	50 × 30	<u>94.0</u>	72.3
He et al. [He et al., 2021]	ResNet	Kinetics400	90.5 × 1	93.7	70.3
MFCD-Net [Battash et al., 2020]	MFNet3D	Kinetics400	128 × 1	93.2	66.9
TEMSN [Li et al., 2021]	ResNet18+ResNet152	ImageNet1K	50.4 × 1	91.8	61.1
MTFD [Guo et al., 2023]	ResNet18	—	<u>25.5 × 1</u>	92.4	**78.7**
MEACI-Net [Li et al., 2022]	I3D-ResNet50	Kinetics400	89.0 × 3	**96.4**	<u>74.4</u>
Proposed	ResNet18-TSM	ImageNet1K	**15.4 × 1**	85.8	58.6
Proposed	ResNet34-TSM	ImageNet1K	28.9 × 1	88.9	61.7
Proposed	ResNet50-TSM	ImageNet1K	33.6 × 1	89.2	62.9

fusion baseline. This result indicated that MIMO pre-training, which trains backbone networks to classify images with MIMO in advance, was essential for a single network to achieve competitive performance compared to the standard late fusion.

4.3.5 Comparison with Previous Methods

We also compared our method with conventional compressed video action recognition methods and summarized the results in Table 4.2. Some conventional methods also focus on efficient compressed video action recognition [Huo et al., 2019, dos Santos et al., 2019]. Still, they explore how to use lightweight networks such as MobileNetV2 [Sandler et al., 2018] and C3D [Tran et al., 2015] as their backbones,

different from our approach that reduces the number of networks to process compressed video features. From Table 4.2, we found that our method requires lower computational complexity than conventional methods, as shown in the GFLOPs × Views column. Our proposed method with ResNet50 still needs lower computational complexity than conventional methods, while some conventional methods use lightweight networks as their backbones. Our method can attain a low computational complexity because it only uses a single network. In contrast, most conventional methods use multiple networks corresponding to each compressed video feature.

In addition, our method was competitive with most conventional methods regarding video classification performance. This result indicates that a single network can reach the same accuracy level as multiple networks specialized in compressed video action recognition.

4.4 Discussion

For the implementation level, our method and early fusion process compressed video features similarly except for output layers for generating predictions. However, our method achieved better video classification performance than early fusion. This section discusses why our method can improve video classification performance against early fusion.

Compared with late fusion, despite its computational complexity, early fusion is not popular because of the performance problem for multi-stream video classification, including compressed video action recognition. Jiang et al. claimed that the reason for the performance problem of early fusion is the high complexity of early fusion input [Jiang et al., 2018]. In multi-stream video classification, inputs for each stream contain different discriminative features and should be effectively fused to improve video classification performance. However, by fusing multi-stream inputs in early layers, these inputs disturb their discriminative features each other. Given the con-

catenated features as inputs like early fusion, networks must first learn how to extract each feature separately from concatenated inputs to fuse these different features effectively. This learning is difficult when we optimize networks to minimize only the cross-entropy loss.

Our method overcomes this problem by forcing networks to internally process multiple inputs separately by the MIMO configuration. Under this training approach, there are no advantages of fusing features to minimize the loss value; therefore, our networks only focus on extracting features from each input. We can improve the video classification performance with a single network by fusing the predictions as output for each stream using separately extracted features.

However, our method still has room for improvement. Recent works show that fusing the multiple features in the intermediate stage of the processing improves multi-stream video classification performance effectively [Ryoo et al., 2019, Ryoo et al., 2020, Feichtenhofer et al., 2019]. In compressed video action recognition, some works already employed this intermediate fusion approach [Li et al., 2020, He et al., 2021]. These previous works construct the intermediate features corresponding to different input features using multiple networks. When considering extending our proposed method to intermediate fusion, we need to address the problem of how to extract the intermediate features. It is a non-trivial problem because they are non-linearly mixed in the intermediate stages of a single network. We need additional modules or loss terms to disentangle our networks' embedding while keeping our method's computational complexity.

4.5 Conclusion

This study presents a novel method for efficient compressed video action recognition. We extend the MIMO approach to compressed video action recognition and construct a single network that acts as late fusion. Our experiments demonstrate that our

method can achieve early fusion-level efficiency in terms of computational complexity and late fusion-level video classification performance. Moreover, our method performs competitive video classification compared to conventional methods while attaining lower computational complexity.

Chapter 5

Conclusion

This thesis tried to reduce the annotation and computational costs of compressed video action recognition.

In Chapter 3, we tackled the annotation cost problem and proposed a novel SSL method named CoVEnPL. This chapter first showed that compressed video features achieved better accuracy than raw RGB frames under the limited labeled data. This result showed the usefulness of compressed video features for the scenario of SSL. Then, we proposed CoVEnPL that makes pseudo labels by aggregating the I-frame, motion vector, and residual predictions. The proposed method improved the accuracy of semi-supervised compressed video action recognition and defeated most RGB frame-based SSL methods in terms of accuracy with a faster inference time.

In Chapter 4, we tackled the computational cost problem and proposed the MussNet model that uses only a single network to process compressed video features simultaneously. We first showed that early fusion is worse than late fusion in compressed video action recognition. Then, we proposed to approximate late fusion using a single network by training the network in the MIMO manner. The proposed model achieved competitive accuracy against the late fusion baseline while keeping the efficiency of a single network. Compared with previous methods, our method achieved one of

the most efficient GFLOPs even when we used a relatively large network, ResNet-50, as the backbone. In addition, the accuracy of the MussNet model was competitive against the previous studies.

We consider that both approaches, SSL and single network-based compressed video action recognition, are promising directions to improve the efficiency of compressed video action recognition. SSL and MIMO have shown remarkable progress in more efficient image processing for images. In addition, compressed video features have their advantage in that different types of features are available without decoding. The author believes combining compressed video action recognition and efficient deep learning methods would lead to more interesting and practical technologies for computer vision tasks, including video classification.

References

[Ballas et al., 2015] Ballas, N., Yao, L., Pal, C., and Courville, A. (2015). Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*.

[Battash et al., 2020] Battash, B., Barad, H., Tang, H., and Bleiweiss, A. (2020). Mimic the raw domain: Accelerating action recognition in the compressed domain. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 684–685.

[Berthelot et al., 2020] Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., and Raffel, C. (2020). Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. In *International Conference on Learning Representations*.

[Berthelot et al., 2019] Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. (2019). Mixmatch: A holistic approach to semi-supervised learning. volume 32.

[Bommes et al., 2020] Bommes, L., Lin, X., and Zhou, J. (2020). Mvmed: Fast multi-object tracking in the compressed domain. In *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 1419–1424. IEEE.

[Boulahia et al., 2021] Boulahia, S. Y., Amamra, A., Madi, M. R., and Daikh, S. (2021). Early, intermediate and late fusion strategies for robust deep learning-based multimodal action recognition. *Machine Vision and Applications*, 32(6):121.

[Carreira et al., 2018] Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C., and Zisserman, A. (2018). A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*.

[Carreira et al., 2019] Carreira, J., Noland, E., Hillier, C., and Zisserman, A. (2019). A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*.

[Carreira and Zisserman, 2017] Carreira, J. and Zisserman, A. (2017). Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308.

[Crasto et al., 2019] Crasto, N., Weinzaepfel, P., Alahari, K., and Schmid, C. (2019). Mars: Motion-augmented rgb stream for action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7882–7891.

[Cubuk et al., 2020] Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703.

[Dalal et al., 2006] Dalal, N., Triggs, B., and Schmid, C. (2006). Human detection using oriented histograms of flow and appearance. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part II 9*, pages 428–441. Springer.

[Dollár et al., 2005] Dollár, P., Rabaud, V., Cottrell, G., and Belongie, S. (2005). Behavior recognition via sparse spatio-temporal features. In *2005 IEEE international workshop on visual surveillance and performance evaluation of tracking and*

surveillance, pages 65–72. IEEE.

[dos Santos et al., 2019] dos Santos, S. F., Sebe, N., and Almeida, J. (2019). Cv-c3d: action recognition on compressed videos with convolutional 3d networks. In *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 24–30. IEEE.

[Duong et al., 2017] Duong, C. T., Lebret, R., and Aberer, K. (2017). Multimodal classification for analysing social media. *arXiv preprint arXiv:1708.02099*.

[Fan et al., 2020] Fan, L., Buch, S., Wang, G., Cao, R., Zhu, Y., Niebles, J. C., and Fei-Fei, L. (2020). Rubiksnet: Learnable 3d-shift for efficient video action recognition. In *European Conference on Computer Vision*, pages 505–521. Springer.

[Feichtenhofer, 2020] Feichtenhofer, C. (2020). X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 203–213.

[Feichtenhofer et al., 2019] Feichtenhofer, C., Fan, H., Malik, J., and He, K. (2019). Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211.

[Feichtenhofer et al., 2016a] Feichtenhofer, C., Pinz, A., and Wildes, R. P. (2016a). Spatiotemporal residual networks for video action recognition. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3476–3484.

[Feichtenhofer et al., 2016b] Feichtenhofer, C., Pinz, A., and Zisserman, A. (2016b). Convolutional two-stream network fusion for video action recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, Los Alamitos, CA, USA. IEEE Computer Society.

[Ferianc and Rodrigues, 2023] Ferianc, M. and Rodrigues, M. (2023). Mimmo: multi-input massive multi-output neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4563–4568.

[Fields, 2019] Fields, T. U. M. R. (2019). Compressed-domain video object tracking using markov random fields with graph cuts optimization. In *Pattern Recognition: 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9-12, 2018, Proceedings*, volume 11269, page 127. Springer.

[Gal and Ghahramani, 2016] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. PMLR.

[Girdhar et al., 2019] Girdhar, R., Tran, D., Torresani, L., and Ramanan, D. (2019). Distinit: Learning video representations without a single labeled video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 852–861.

[Grauman et al., 2022] Grauman, K., Westbury, A., Byrne, E., Chavis, Z., Furnari, A., Girdhar, R., Hamburger, J., Jiang, H., Liu, M., Liu, X., et al. (2022). Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012.

[Guo et al., 2023] Guo, J., Zhang, J., Li, S., Zhang, X., and Ma, M. (2023). Mtfd: Multi-teacher fusion distillation for compressed video action recognition. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

[Han et al., 2016] Han, S., Pool, J., Narang, S., Mao, H., Tang, S., Elsen, E., Catanzaro, B., Tran, J., and Dally, W. J. (2016). Dsd: Regularizing deep neural networks with dense-sparse-dense training flow. *arXiv preprint arXiv:1607.04381*.

[Han et al., 2015] Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. volume 28.

[Hara et al., 2017] Hara, K., Kataoka, H., and Satoh, Y. (2017). Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings*

of the IEEE international conference on computer vision workshops, pages 3154–3160.

[Havasi et al., 2021] Havasi, M., Jenatton, R., Fort, S., Liu, J. Z., Snoek, J., Lakshminarayanan, B., Dai, A. M., and Tran, D. (2021). Training independent subnetworks for robust prediction. In *International Conference on Learning Representations*.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[He et al., 2021] He, L., Zhang, M., Zhang, S., and Li, F. (2021). Physical knowledge driven multi-scale temporal receptive field network for compressed video action recognition. In *Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers*, pages 625–630.

[Hinton et al., 2015] Hinton, G., Vinyals, O., Dean, J., et al. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[Horton et al., 2023] Horton, M., Mehta, S., Farhadi, A., and Rastegari, M. (2023). Bytes are all you need: Transformers operating directly on file bytes. *arXiv preprint arXiv:2306.00238*.

[Hu et al., 2019] Hu, D., Wang, C., Nie, F., and Li, X. (2019). Dense multimodal fusion for hierarchically joint representation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3941–3945. IEEE.

[Huang et al., 2017] Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. (2017). Snapshot ensembles: Train 1, get m for free.

[Huo et al., 2019] Huo, Y., Xu, X., Lu, Y., Niu, Y., Lu, Z., and Wen, J.-R. (2019). Mobile video action recognition. *arXiv preprint arXiv:1908.10155*.

[Jiang et al., 2018] Jiang, H., Li, Y., Song, S., and Liu, J. (2018). Rethinking fusion

baselines for multi-modal human action recognition. In *Pacific Rim Conference on Multimedia*, pages 178–187. Springer.

[Jin et al., 2016] Jin, X., Yuan, X., Feng, J., and Yan, S. (2016). Training skinny deep neural networks with iterative hard thresholding methods. *arXiv preprint arXiv:1607.05423*.

[Jing et al., 2021] Jing, L., Parag, T., Wu, Z., Tian, Y., and Wang, H. (2021). Videossl: Semi-supervised learning for video classification.

[Joze et al., 2020] Joze, H. R. V., Shaban, A., Iuzzolino, M. L., and Koishida, K. (2020). Mmtm: Multimodal transfer module for cnn fusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13289–13299.

[Karpathy et al., 2014] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Klaser et al., 2008] Klaser, A., Marszałek, M., and Schmid, C. (2008). A spatio-temporal descriptor based on 3d-gradients. In *BMVC 2008-19th British Machine Vision Conference*, pages 275–1. British Machine Vision Association.

[Kondratyuk et al., 2021] Kondratyuk, D., Yuan, L., Li, Y., Zhang, L., Tan, M., Brown, M., and Gong, B. (2021). Movinets: Mobile video networks for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16020–16030.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[Kuehne et al., 2011] Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. (2011). Hmdb: A large video database for human motion recognition. In *2011 International conference on computer vision*, pages 2556–2563. IEEE.

[Laine and Aila, 2016] Laine, S. and Aila, T. (2016). Temporal ensembling for semi-supervised learning.

[Laptev and Lindeberg, 2003] Laptev, I. and Lindeberg, T. (2003). Space-time interest points. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 432–439 vol.1.

[Le Gall, 1991] Le Gall, D. (1991). Mpeg: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58.

[Lee et al., 2013] Lee, D.-H. et al. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896.

[Li et al., 2022] Li, B., Chen, J., Zhang, D., Bao, X., and Huang, D. (2022). Representation learning for compressed video action recognition via attentive cross-modal interaction with motion enhancement.

[Li et al., 2021] Li, B., Kong, L., Zhang, D., Bao, X., Huang, D., and Wang, Y. (2021). Towards practical compressed video action recognition: a temporal enhanced multi-stream network. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3744–3750. IEEE.

[Li et al., 2020] Li, J., Wei, P., Zhang, Y., and Zheng, N. (2020). A slow-i-fast-p architecture for compressed video action recognition. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2039–2047.

[Lin et al., 2019] Lin, J., Gan, C., and Han, S. (2019). Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7083–7093.

[Liu et al., 2021a] Liu, X., Jin, L., Han, X., Lu, J., You, J., and Kong, L. (2021a).

Identity-aware facial expression recognition in compressed video. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 7508–7514. IEEE.

[Liu et al., 2021b] Liu, Z., Wang, L., Wu, W., Qian, C., and Lu, T. (2021b). Tam: Temporal adaptive module for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13708–13718.

[Micikevicius et al., 2017] Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al. (2017). Mixed precision training. *arXiv preprint arXiv:1710.03740*.

[Miyato et al., 2018] Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

[Murahari et al., 2022] Murahari, V., Jimenez, C., Yang, R., and Narasimhan, K. (2022). Datamux: data multiplexing for neural networks. *Advances in Neural Information Processing Systems*, 35:17515–17527.

[Ng et al., 2018] Ng, J. Y.-H., Choi, J., Neumann, J., and Davis, L. S. (2018). Actionflownet: Learning motion representation for action recognition. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1616–1624. IEEE.

[Oliver et al., 2018] Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., and Goodfellow, I. (2018). Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[Park and Johnson, 2023] Park, J. and Johnson, J. (2023). Rgb no more: Minimally-decoded jpeg vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22334–22346.

[Park et al., 2023] Park, S., Chun, S., Heo, B., Kim, W., and Yun, S. (2023). Seit:

Storage-efficient vision training with tokens using 1% of pixel storage. *arXiv preprint arXiv:2303.11114*.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. volume 32.

[Piergiovanni et al., 2022] Piergiovanni, A., Angelova, A., and Ryoo, M. S. (2022). Tiny video networks. *Applied AI Letters*, 3(1):e38.

[Ramé et al., 2021] Ramé, A., Sun, R., and Cord, M. (2021). Mixmo: Mixing multiple inputs for multiple outputs via deep subnetworks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 823–833.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

[Ryoo et al., 2020] Ryoo, M. S., Piergiovanni, A., Kangaspunta, J., and Angelova, A. (2020). Assemblenet++: Assembling modality representations via attention connections. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 654–671. Springer.

[Ryoo et al., 2019] Ryoo, M. S., Piergiovanni, A., Tan, M., and Angelova, A. (2019). Assemblenet: Searching for multi-stream neural connectivity in video architectures.

[Sadanand and Corso, 2012] Sadanand, S. and Corso, J. J. (2012). Action bank: A high-level representation of activity in video. In *2012 IEEE Conference on computer vision and pattern recognition*, pages 1234–1241. IEEE.

[Sandler et al., 2018] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

[Scovanner et al., 2007] Scovanner, P., Ali, S., and Shah, M. (2007). A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 357–360.

[Sevilla-Lara et al., 2018] Sevilla-Lara, L., Liao, Y., Güney, F., Jampani, V., Geiger, A., and Black, M. J. (2018). On the integration of optical flow and action recognition. In *German conference on pattern recognition*, pages 281–297. Springer.

[Sharma et al., 2015] Sharma, S., Kiros, R., and Salakhutdinov, R. (2015). Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*.

[Shou et al., 2019] Shou, Z., Lin, X., Kalantidis, Y., Sevilla-Lara, L., Rohrbach, M., Chang, S.-F., and Yan, Z. (2019). Dmc-net: Generating discriminative motion cues for fast compressed video action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1268–1277.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. volume 27.

[Sohn et al., 2020] Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., Cubuk, E. D., Kurakin, A., and Li, C.-L. (2020). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems*, volume 33, pages 596–608. Curran Associates, Inc.

[Soomro et al., 2012] Soomro, K., Zamir, A. R., and Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

[Stroud et al., 2020] Stroud, J., Ross, D., Sun, C., Deng, J., and Sukthankar, R. (2020). D3d: Distilled 3d networks for video action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 625–634.

[Sun et al., 2022] Sun, R., Ramé, A., Masson, C., Thome, N., and Cord, M. (2022). Towards efficient feature sharing in mimo architectures. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2697–2701.

[Sutskever et al., 2013] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[Tarvainen and Valpola, 2017] Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[Tran et al., 2015] Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497.

[Tran et al., 2018] Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., and Paluri, M. (2018). A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459.

[Vapnik, 1963] Vapnik, V. N. (1963). Pattern recognition using generalized portrait method. *Automation and remote control*, 24(6):774–780.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Wang et al., 2013] Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. (2013). Dense trajectories and motion boundary descriptors for action recognition. *International*

Journal of Computer Vision, 103:60–79.

[Wang and Schmid, 2013] Wang, H. and Schmid, C. (2013). Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision*, pages 3551–3558.

[Wang et al., 2014] Wang, L., Qiao, Y., and Tang, X. (2014). Video action detection with relational dynamic-poselets. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 565–580. Springer.

[Wang et al., 2018] Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Van Gool, L. (2018). Temporal segment networks for action recognition in videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(11):2740–2755.

[Wang et al., 2019] Wang, S., Lu, H., and Deng, Z. (2019). Fast object detection in compressed video. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7104–7113.

[Wiles et al., 2022] Wiles, O., Carreira, J., Barr, I., Zisserman, A., and Malinowski, M. (2022). Compressed vision for efficient video understanding. In *Proceedings of the Asian Conference on Computer Vision*, pages 4581–4597.

[Willems et al., 2008] Willems, G., Tuytelaars, T., and Van Gool, L. (2008). An efficient dense and scale-invariant spatio-temporal interest point detector. In *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part II 10*, pages 650–663. Springer.

[Wu et al., 2018] Wu, C.-Y., Zaheer, M., Hu, H., Manmatha, R., Smola, A. J., and Krähenbühl, P. (2018). Compressed video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6026–6035.

[Wu et al., 2019] Wu, M.-C., Chiu, C.-T., and Wu, K.-H. (2019). Multi-teacher knowledge distillation for compressed video action recognition on deep neural net-

works. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2202–2206. IEEE.

[Xiao et al., 2021] Xiao, J., Jing, L., Zhang, L., He, J., She, Q., Zhou, Z., Yuille, A., and Li, Y. (2021). Learning from temporal gradient for semi-supervised action recognition. *arXiv preprint arXiv:2111.13241*.

[Xie et al., 2020] Xie, Q., Dai, Z., Hovy, E., Luong, T., and Le, Q. (2020). Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Curran Associates, Inc.

[Xie et al., 2018] Xie, S., Sun, C., Huang, J., Tu, Z., and Murphy, K. (2018). Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[Xiong et al., 2021] Xiong, B., Fan, H., Grauman, K., and Feichtenhofer, C. (2021). Multiview pseudo-labeling for semi-supervised learning from video. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7209–7219.

[Xu et al., 2021] Xu, Y., Wei, F., Sun, X., Yang, C., Shen, Y., Dai, B., Zhou, B., and Lin, S. (2021). Cross-model pseudo-labeling for semi-supervised action recognition. *arXiv preprint arXiv:2112.09690*.

[Yalniz et al., 2019] Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M., and Mahajan, D. (2019). Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*.

[Yue-Hei Ng et al., 2015] Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., and Toderici, G. (2015). Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702.

[Yun et al., 2019] Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer*

vision, pages 6023–6032.

[Zhai et al., 2019] Zhai, X., Oliver, A., Kolesnikov, A., and Beyer, L. (2019). S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1476–1485.

[Zhang et al., 2016] Zhang, B., Wang, L., Wang, Z., Qiao, Y., and Wang, H. (2016). Real-time action recognition with enhanced motion vector cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2718–2726.

[Zhang et al., 2018] Zhang, B., Wang, L., Wang, Z., Qiao, Y., and Wang, H. (2018). Real-time action recognition with deeply transferred motion vector cnns. *IEEE Transactions on Image Processing*, 27(5):2326–2339.

[Zhang et al., 2021] Zhang, B., Wang, Y., Hou, W., Wu, H., Wang, J., Okumura, M., and Shinozaki, T. (2021). Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems*, 34.

[Zhang et al., 2017] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

[Zhu et al., 2017] Zhu, Y., Lan, Z., Newsam, S., and Hauptmann, A. G. (2017). Hidden Two-Stream Convolutional Networks for Action Recognition. *arXiv preprint arXiv:1704.00389*.

[Zou et al., 2021] Zou, Y., Choi, J., Wang, Q., and Huang, J.-B. (2021). Learning representational invariances for data-efficient action recognition. *arXiv preprint arXiv:2103.16565*.