ABSTRACT

Title of Dissertation: LONG-TERM TEMPORAL MODELING FOR VIDEO ACTION UNDERSTANDING

Xitong Yang
Doctor of Philosophy, 2021

Dissertation Directed by: Professor Larry S. Davis
Department of Computer Science

The tremendous growth in video data, both on the internet and in real life, has encouraged the development of intelligent systems that can automatically analyze video contents and understand human actions. Therefore, video understanding has been one of the fundamental research topics in computer vision. Encouraged by the success of deep neural networks on image classification, many efforts have been made in recent years to extend the deep networks to video understanding. However, new challenges arise when the temporal characteristic of videos is taken into account. In this dissertation, we study two long-standing problems that play important roles in effective temporal modeling in videos: (i) How to extract motion information from raw video frames? (ii) How to capture long-range dependencies in time and model their temporal dynamics?

To address the above issues, we first introduce hierarchical contrastive motion learning, a novel self-supervised learning framework to extract effective motion representations from raw video frames. Our approach progressively learns a hierarchy of motion features, from low-level pixel movements to higher-level semantic dynamics, in a fully

self-supervised manner. Next, we investigate the self-attention mechanism for long-range temporal modeling, and demonstrate that the entangled modeling of spatio-temporal information fails to capture temporal relationships among frames explicitly. To this end, we propose Global Temporal Attention (GTA), which performs global temporal attention on top of spatial attention in a decoupled manner. Unlike conventional self-attention that computes an instance-specific attention matrix, GTA directly learns a global attention matrix that is intended to encode temporal structures that generalize across different samples.

While the performance of video action recognition has been significantly improved by the aforementioned methods, they are still restricted to model temporal information within short clips. To overcome this limitation, we introduce a collaborative memory mechanism that encodes information across multiple sampled clips of a video at each training iteration. Our proposed framework is end-to-end trainable and significantly improves the accuracy of video classification at a negligible computational overhead. Finally, we present a spatio-temporal progressive learning framework (STEP) for spatio-temporal action detection. Our approach performs a multi-step optimization process that progressively refines the initial proposals towards the final solution. In this way, our approach can effectively make use of long-term temporal information by handling the spatial displacement problem in long action tubes.

LONG-TERM TEMPORAL MODELING FOR VIDEO ACTION UNDERSTANDING

by

Xitong Yang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Professor Larry S. Davis, Chair/Advisor
Professor David Jacobs
Professor Min Wu
Professor Abhinav Shrivastava
Professor Furong Huang

Dedication

To my dearest wife and parents.

Acknowledgments

First and foremost I would like to thank my advisor, Professor Larry Davis for giving me an invaluable opportunity to work on computer vision, one of the most excitingly progressing fields in computer science. I really appreciate his guidance, encouragement and offering great freedom to pursue the research direction I am interested in. Secondly, I would like to express my gratitude to Professor David Jacobs, Professor Min Wu, Professor Abhinav Shrivastava and Professor Furong Huang for serving as my dissertation committee.

I would also like to thank all my mentors and collaborators in academia and industry: Professor Jiebo Luo, Dr. Xiaodong Yang, Dr. Sifei Liu, Dr. Heng Wang, Professor Lorenzo Torresani, Haoqi Fan, Dr. Yi-Ting Chen, Dr. Teruhisa Misu for their help and guidance.

I am grateful to be able to work with the amazing colleagues at my lab: Zuxuan Wu, Peng Zhou, Xiyang Dai, Xintong Han, Zhe Wu, Hengduo Li, Mingfei Gao, Ruichi Yu, Ahmed Taha. And I will be always thankful to my friends at VIStA who enlighten the very beginning of my research career: Tianran Hu, Quanzeng You, Yuncheng Li.

Last but not least, I have to give my greatest thanks to my family: my mother Miaoying Guan and my father Jinqiang Yang, who raise me, love me and always support me. I thank my wife, Jiani Yang, for her company, support, trust, love and the happiness

she has been giving me over the past 13 years.

It is impossible to remember all those who have helped me, and I apologize to those I've carelessly left out.

Table of Contents

List of Tables

List of Figures

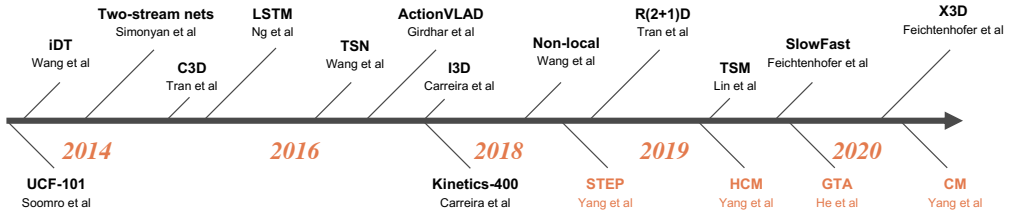
Chapter 1: Introduction

1.1 Motivation

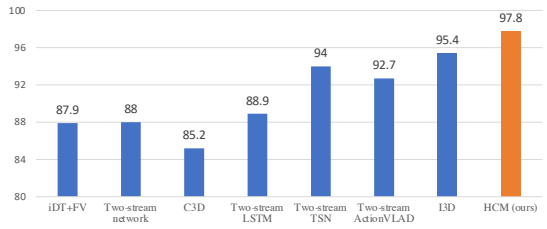
Recent years have witnessed a tremendous growth in video data: on the internet, more than 75% of all IP traffic comes from video streaming and downloads [1]; in real life, millions of cameras (such as surveillance cameras and in-vehicle cameras) are in operation for service or security reasons. Therefore, there is an urgent need to develop intelligent systems that can automatically analyze video contents and understand human actions in videos. For example, online video-sharing websites need to understanding actions and events in each video to provide accurate recommendations and advertisement; social media platforms require automatic analysis of the video posts to guarantee their integrity and identify viral videos that include harmful contents.

Compared with still image analysis, video understanding is much more challenging due to the introduction of the temporal component in videos. To show the importance of temporal modeling, let us take a brief overview of some representative work in video action recognition, as shown in Figure 1.1. Dating back to 2013, where deep learning was not yet the mainstream algorithm for computer vision, researchers were more focused on designing hand-crafted motion features for video understanding, such as dense

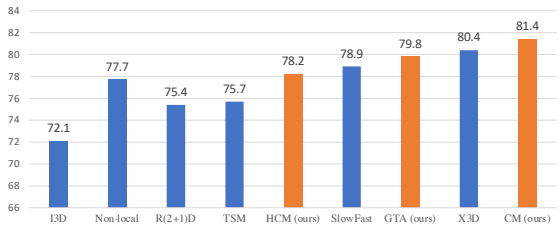
[1] Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper



(a) A Chronological overview of recent representative work in video action recognition.



(b) UCF-101 (13.3K videos, 101 action classes)



(c) Kinetics-400 (260K videos, 400 action classes)

Figure 1.1: An overview of recent representative work in video action recognition and their performance on UCF-101 and Kinetics-400. Our works introduced in this dissertation are highlighted in orange.

trajectory (iDT) [1] and 3D-SIFT [2]. When convolutional neural networks (CNNs), like AlexNet [3], had been attracting more and more attention, people started applying deep neural networks to videos as well. One seminal work is Two-stream Network [4], which involves a spatial stream that models appearance information using a single frame, and a temporal stream that models motion information using optical flow as input. There are also other attempts that use recurrent networks [5] or extend the 2D spatial kernels to 3D spatio-temporal kernels [6] in order to jointly model the spatial and temporal information in video frames. From then on, more and more temporal modeling methods are proposed and achieve continuous improvement, such as Temporal Segment Network [7], ActionVLAD [8], Inflated 3D Network [9], Non-local [10], Temporal Shift Moudle [11], SlowFast [12], *etc*. We observe two trends here: (i) The use of hand-crafted features is less favorable in recent work, (ii) Modeling long-range temporal dependencies has been attracting more and more attention. It also shows that effective temporal modeling is

challenging – unlike 2D convolution in still image analysis, there is still no predominant method for temporal modeling in video understanding.

In this dissertation, we study two long-standing problems that play important roles in effective temporal modeling: (i) How to extract motion information from raw video frames? (ii) How to capture long-range dependencies in time and model their temporal dynamics? Challenges of these problems stem from the temporal characteristic of video data, as well as the large scale of video models and datasets in practice. For example, "ground truth motion" of a video is usually unavailable for large-scale video datasets because it requires fine-grained annotations on a large number of videos each with hundreds frames. Even with the help of automatic motion extraction algorithms (*e.g.*, TV-L1 [13] for optical flow estimation), computing and storing the motion features are still extremely time-consuming and require large storage space. In addition, the large number of frames in a video also pose challenges for the optimization of video models. Due to the high computational cost and GPU memory constraints, most video models are optimized over short clips sampled from the full-length video, which actually impedes their ability to learn long-term temporal dependencies.

1.2 Our Approaches

To address the above issues, this dissertation presents multiple approaches for video understanding that focuses on extracting motion information and modeling long-range temporal dependencies. We first propose a self-supervised learning framework to learn hierarchical motion representation from raw video frames. We then present a global tem-

poral attention module that advances the standard attention mechanism for effective temporal modeling. After that, we present an end-to-end video-level learning framework that is capable of capturing temporal dependencies beyond short clips. Finally, we introduce a novel progressive learning framework for spatio-temporal action detection. Our works keep pushing the state-of-the-art at the time if we put them back into the chronological overview in Figure 1.1 (highlighted in orange).

1.2.1 Hierarchical Contrastive Motion Learning for Video Action Recognition

One central question for video action recognition is how to model motion. In the first work, we present hierarchical contrastive motion learning, a novel self-supervised learning framework to extract effective motion representations from raw video frames. Our approach progressively learns a hierarchy of motion features that correspond to different abstraction levels in a network. This hierarchical design bridges the semantic gap between low-level movement cues and high-level recognition tasks, and promotes the fusion of appearance and motion information at multiple levels. At each level, an explicit motion self-supervision is provided via contrastive learning to enforce the motion features at the current level to predict the future ones at the previous level. Thus, the motion features at higher levels are trained to gradually capture semantic dynamics and evolve more discriminative for video action recognition. Our motion learning module is lightweight and flexible to be embedded into various backbone networks. Extensive experiments on four benchmarks show that the proposed approach consistently achieves superior results.

1.2.2 Global Temporal Attention for Video Action Understanding

Self-attention learns pairwise interactions to model long-range dependencies, yielding great improvements for video action recognition. In the second work, we seek a deeper understanding of self-attention for temporal modeling in videos. We first demonstrate that the entangled modeling of spatio-temporal information by flattening all pixels is sub-optimal, failing to capture temporal relationships among frames explicitly. To this end, we introduce Global Temporal Attention (GTA), which performs global temporal attention on top of spatial attention in a decoupled manner. We apply GTA on both pixels and semantically similar regions to capture temporal relationships at different levels of spatial granularity. Unlike conventional self-attention that computes an instance-specific attention matrix, GTA directly learns a global attention matrix that is intended to encode temporal structures that generalize across different samples. We further augment GTA with a cross-channel multi-head fashion to exploit channel interactions for better temporal modeling. Extensive experiments on 2D and 3D networks demonstrate that our approach consistently enhances temporal modeling and provides state-of-the-art performance on three video action recognition datasets.

1.2.3 Beyond Short Clips: End-to-End Video-Level Learning with Collaborative Memories

The standard way of training video models entails sampling at each iteration a single clip from a video and optimizing the clip prediction with respect to the video-level

label. In the third work, we argue that a single clip may not have enough temporal coverage to exhibit the label to recognize, since video datasets are often weakly labeled with categorical information but without dense temporal annotations. Furthermore, optimizing the model over brief clips impedes its ability to learn long-term temporal dependencies.

To overcome these limitations, we introduce a collaborative memory mechanism that encodes information across multiple sampled clips of a video at each training iteration. This enables the learning of long-range dependencies beyond a single clip. We explore different design choices for the collaborative memory to ease the optimization difficulties. Our proposed framework is end-to-end trainable and significantly improves the accuracy of video classification at a negligible computational overhead. Through extensive experiments, we demonstrate that our framework generalizes to different video architectures and tasks, outperforming the state of the art on both action recognition (*e.g.*, Kinetics-400 & 700, Charades, Something-Something-V1) and action detection (*e.g.*, AVA v2.1 & v2.2).

1.2.4 Spatio-Temporal Progressive Learning for Video Action Detection

In the final work, we focus on improving long-term temporal modeling for spatio-temporal video action detection. Conventional approaches for action detection are mostly extended from the image object detectors by performing detection at a clip (i.e., a short video snippet) level. In particular, 2D region proposals are first generated at the center frame of the video clip and then replicated over time to get 3D action proposals. However, this method could be problematic if the action tubes involve large spatial displacement due

to long sequences or rapid movement of actors or camera.

Therefore, we propose a novel progressive learning framework (STEP) to tackle this problem. Rather than directly detecting actions all in one run, we perform a multi-step optimization process that progressively refines the initial proposals (coarse-scale short-range cuboids) towards the final solution. In this way, high-quality proposals (i.e., adhere to action movements) can be gradually obtained at later steps by leveraging the regression outputs from previous steps. At each step, we adaptively extend the proposals in time to incorporate more related temporal context. Compared to the prior work that performs action detection in one run, our progressive learning framework is able to naturally handle the spatial displacement within action tubes and therefore provides a more effective way for spatio-temporal modeling. We extensively evaluate our approach on UCF101 and AVA, and demonstrate superior detection results. Remarkably, we achieve mAP of 75.0% and 18.6% on the two datasets with 3 progressive steps and using respectively only 11 and 34 initial proposals.

1.3 Organization

The remainder of this dissertation is organized as follows. Chapter 2 introduces the hierarchical contrastive learning framework for motion representation learning. Chapter 3 presents a global temporal attention module for action recognition. Chapter 4 presents collaborative memory, an end-to-end framework for video-level learning. Chapter 5 introduces a progressive learning framework for spatio-temporal action detection. Finally, in Chapter 6, we conclude and discuss future research directions.

Chapter 2: Hierarchical Contrastive Motion Learning for Video Action Recognition

2.1 Introduction

Motion provides abundant and powerful cues for understanding the dynamic visual world. A broad range of video understanding tasks can benefit from the introduction of motion information, such as action recognition [1, 4], activity detection [14, 15], object tracking [16, 17], etc. Thus, how to extract and model temporal motion is one of the fundamental problems in video understanding. While early methods in this field mostly rely on the off-the-shelf pre-computed motion features such as optical flow, recent works have been actively exploiting convolutional neural networks (CNNs) for more effective motion learning from raw video frames [18, 19], encouraged by the tremendous success of end-to-end learning in various vision tasks.

A key challenge for end-to-end motion representation learning is to design an effective supervision. Unlike many other tasks that afford plenty of well-defined annotations, "ground truth motion" is usually unavailable or even undefined for motion learning in practice. One popular idea is to extract motion features by means of the action recognition supervision. However, the classification loss is shown to be sub-optimal in this task

as it only provides an implied supervision to guide motion learning [20]. This supervision is also prone to be biased towards appearance information since some video action benchmarks can be mostly solved by considering static images without temporal modeling [21]. Recently, some efforts have been made to explore pretext tasks with direct supervisions for motion learning, such as optical flow prediction [18] and video frame reconstruction [22]. Although having shown promising results, such supervisions are restricted to pixel-wise and short-term motion as they hinge on pixel photometric loss and movement between adjacent frames.

In light of the above observations, we introduce a novel self-supervised learning framework that enables more explicit motion supervision at multiple feature abstraction levels, which we term hierarchical contrastive motion learning. Specifically, given preliminary motion cues as a bootstrap, our approach progressively learns a hierarchy of motion features in a bottom-up manner, as illustrated in Figure 2.1(a). This hierarchical design is proposed to bridge the semantic gap between the low-level preliminary motion and the high-level recognition task—analogous to the findings in neuroscience that humans perceive motion patterns in a hierarchical way [23, 24]. At each level, a discriminative contrastive loss [25, 26] is used to provide an explicit self-supervision to enforce the motion features at current level to predict the future ones at previous level. In contrast to the aforementioned pretext tasks that focus on low-level image details, the contrastive learning encourages the model to learn useful semantic dynamics from previously learned motion features at a lower level, and is more favorable for motion learning at higher levels where the spatial and temporal resolutions of feature maps are low [27–29]. To acquire the preliminary motion cues to initialize the hierarchical motion learning, we exploit the

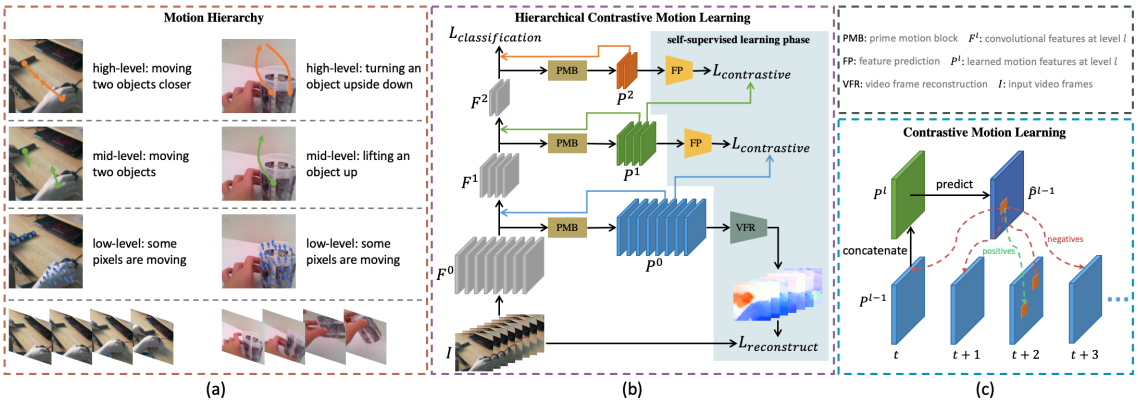


Figure 2.1: (a) Illustration of the hierarchy of progressively learned motion cues: from pixel-level and short-range movement to semantic temporal dynamics. (b) Architecture of the proposed motion learning module embedded in a backbone network. (c) Overview of positive and negative sampling for contrastive learning to adopt higher-level motion features to predict future ones at lower-level.

video frame reconstruction [30, 31] as an auxiliary task such that the whole motion representation learning enjoys an unified self-supervised setup.

The proposed motion learning module is realized via a side network branch, which is lightweight and flexible to embed into a variety of backbone CNNs. Our hierarchical design promotes appearance and motion fusion by integrating the learned motion features into the backbone network at multiple abstraction levels [32]. Such a multi-level fusion paradigm is unachievable for previous motion learning methods [22, 33] that depend solely on low-level motion supervisions. It is also noteworthy that our approach only introduces a small overhead to the computation of a backbone network at inference time. As shown in Figure 2.1(b), the side branch (i.e., the shaded region) for self-supervised motion learning is discarded after training, and only the learned motion features are retained in the form of residual connections [34]. Compared to the two-stream method [4] requiring an additional temporal stream on the pre-computed optical flow, our approach is capable of boosting action recognition with a minor computational increase.

We summarize our main contributions as follows. First, we propose a new learning framework for motion representation learning from raw video frames. Second, we advance contrastive learning to a hierarchical design that bridges the semantic gap between low-level motion cues and high-level recognition tasks. Third, to our knowledge, this work provides the first attempt to empower contrastive learning in motion representation learning for large-scale video action recognition. Fourth, our approach achieves superior results on four benchmarks without relying on off-the-shelf motion features or supervised pre-training.

2.2 Related Work

Motion Extraction. A large family of video action research focuses on motion modeling. The two-stream networks operate on low-level motion inputs such as optical flow or even frame differences [35, 36]. However, the short-range two-frame motion is ineffective at capturing high-level action semantics. It is also popular to use 3D CNNs [37, 38] and RNNs [5, 39] to simultaneously model appearance and motion cues and capture long-term temporal context.

Recently, SlowFast [12] is introduced to extract motion through a fast pathway at a fine temporal resolution and a slow pathway to capture appearance at a low frame rate. STM [40] proposes channel-wise spatio-temporal and motion modules to enhance motion feature encoding. ActionFlowNet [18] uses pre-computed optical flow as an additional supervision to encode motion together with appearance in a single stream network. To integrate optical flow with action networks, a differentiable representation flow layer is

developed in [41]. By formulating the TV-L1 algorithm [13] in a customized network layer, TVNet [42] produces optical flow like motion features to complement static appearance. Inspired by the correlation layer in FlowNet [43], wang2020video [19] and MotionSqueeze [44] leverage the correlation operation to extract motion from convolutional features. All these studies employ classification loss as an indirect supervision or mimic optical flow design to learn motion extraction, while in this work we aim to explicitly realize motion learning through the proposed hierarchical contrastive learning in a self-supervised way.

Self-Supervised Learning. To take advantage of the abundant unlabeled videos, numerous methods have been developed for video representation learning by different self-supervisory signals, such as frame interpolation [33, 45–47], sequence ordering [48–50], future prediction [51–54]. While these methods do not require any pre-trained networks nor video annotations, the learned models tend to focus on low-level information and may not effectively catch the high-level and long-term temporal dynamics.

Contrastive learning has been recently explored for self-supervised learning of video representations [29, 55–59]. Several pretext tasks, such as future prediction [29, 56] and augmentation invariance [55, 59], are designed to facilitate learning of spatio-temporal representations that can benefit high-level video recognition tasks. In contrast to the existing work that focuses on self-supervised learning or model pretraining, our work leverages contrastive learning to extract motion information with an overall goal to improve supervised video classification. Moreover, we conduct contrastive motion learning at multiple levels across the hierarchical structure of a backbone network and present a progressive training strategy to build the higher-level motion from the well-learned lower-level ones.

2.3 Method

Our approach has two phases: a self-supervised learning phase for hierarchical contrastive motion learning from raw video frames, and a joint learning phase that leverages the learned motion features to improve action recognition. In this section, we describe in details the two learning phases.

2.3.1 Formulation

We denote the convolutional features at different levels of a backbone network as $\{\mathcal{F}^0, ..., \mathcal{F}^{L-1}\}$, where L is the number of abstraction levels. Our goal is to learn a hierarchy of motion representations $\{\mathcal{P}^0, ..., \mathcal{P}^{L-1}\}$ that correspond to the different levels. As the first step, we employ video frame reconstruction to obtain the preliminary motion cues $\mathcal{P}^0$, which function as a bootstrap for the following hierarchical motion learning. With that, we progressively learn the motion features in a bottom-up manner. At each level $l > 0$, we learn the motion features $\mathcal{P}^l$ by enforcing them to predict the future motion features at the previous level $l - 1$, as described in Sec. 2.3.2. We use the contrastive loss as an objective so that $\mathcal{P}^l$ is trained to capture semantic temporal dynamics from $\mathcal{P}^{l-1}$. The learned motion features at each level are integrated into a backbone network via residual connections to perform appearance and motion feature fusion: $\mathcal{Z}^l = \mathcal{F}^l + g^l(\mathcal{P}^l)$, where $g^l(\cdot)$ is used to match the feature dimensions. After learning motion at all levels, we jointly train the whole network for action recognition in a multi-tasking manner, as presented in Sec. 2.3.4.

2.3.2 Self-Supervised Motion Learning

Prime Motion Block. We first introduce a lightweight prime motion block to transform the convolutional features of a backbone network to more discriminative representations for motion learning. The key component of this block is a cost volume layer, which is inspired by the success of using cost volumes in stereo matching [60] and optical flow estimation [61]. A cost volume is initially used to store the costs that measure how well a pixel in one frame matches other pixels in another frame to catch the inter-frame pixel-wise relations that indicate the rough motion.

In our case, given a sequence of convolutional features $\mathcal{F} = \{F_0, ..., F_{T-1}\}$ with length T, we first conduct an $1 \times 1 \times 1$ convolution to reduce the input channels by $1/\beta$, denoted as $\tilde{\mathcal{F}}$. This operation significantly reduces the computational overhead of prime motion block, and provides more compact representations to reserve the essential information to compute cost volumes. The adjacent features are then re-organized to feature pairs $\tilde{\mathcal{F}}^* = \left\{(\tilde{F}_0, \tilde{F}_1), ..., (\tilde{F}_{T-2}, \tilde{F}_{T-1}), (\tilde{F}_{T-1}, \tilde{F}_{T-1})\right\}$, which is used to construct the cost volumes. The matching cost between two features is defined as:

$$\text{cv}_t(x_1, y_1, x_2, y_2) = \text{sim}(\tilde{F}_t(x_1, y_1), \tilde{F}_{t+1}(x_2, y_2)), \tag{2.1}$$

where $\tilde{F}_t(x, y)$ denotes the feature vector at time t and position (x, y), and the cosine distance is used as the similarity function: $\text{sim}(u, v) = u^T v / \|u\| \|v\|$. Note that we replicate the last feature map $\tilde{F}_{T-1}$ to compute their cost volume in order to keep the original temporal resolution.

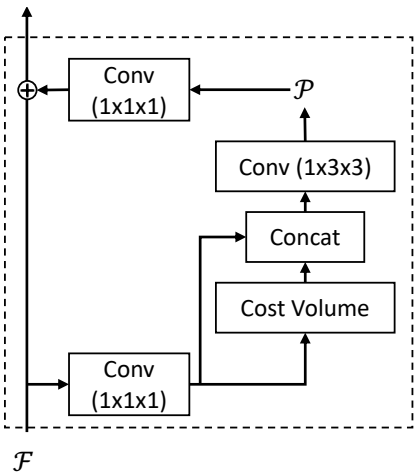


Figure 2.2: Architecture of the prime motion block. $\mathcal{F}$ and $\mathcal{P}$ denote the convolutional feature from backbone network and the extracted motion feature, respectively.

While constructing a full cost volume over the whole feature map is computationally expensive, we construct a "partial" cost volume following the practice in [62]. So we limit the search range with the max displacement of (x_2, y_2) to be d and use a striding factor s to handle large displacements without increasing the computation. As a result, the cost volume layer outputs a feature tensor of size $M \times H \times W$, where $M = (2 \times \lfloor d/s \rfloor + 1)^2$ and H, W denote the height and width of a feature map. It is noteworthy that computing cost volumes is lightweight as it has no learnable parameters and much fewer FLOPs than 3D convolutions.

Finally, we combine the cost volumes with the features obtained after dimension reduction, motivated by the observation that the two features provide complementary information for localizing motion boundaries. We concatenate the two features in channels and then perform a 2D convolution for information fusion. The whole prime motion block is wrapped as a residual block [34] such that the motion features $\mathcal{P}$ can be inserted into a backbone network seamlessly, as illustrated in Figure 2.2. Batch normalization [63] and ReLU are applied after each convolutional layer and cost volume layer.

Hierarchical Motion Learning. Although the prime motion block extracts rough motion features from convolutional features, we find that such features are easily biased towards appearance when jointly trained with the backbone network. Thus, an explicit motion supervision is vital for more effective motion learning at each level.

We propose a multi-level self-supervised objective based on the contrastive loss [25, 26, 64], inspired by the recent success of contrastive predictive coding for self-supervised representation learning [27, 29, 65]. Our goal is to employ the higher-level motion features as a conditional input to guide the prediction of the future lower-level motion features that are well-learned from a previous step. By this way, the higher-level features are forced to understand a more abstract trajectory that summarizes motion dynamics from the lower-level ones. This objective therefore allows us to extract slowly varying features that progressively correspond to high-level semantic concepts [27, 29, 66].

Formally, let us denote the motion features generated by the prime motion block at level $l > 0$ as $\mathcal{P}^l = \left\{P_0^l, ..., P_{T^l-1}^l\right\}$, where T^l indicates the sequence length at this level. In order to train $\mathcal{P}^l$, we enforce P_t^l to predict the future motion features at the previous level (i.e., $P_{>t}^{l-1}$), conditioned on the motion feature at the start time P_t^{l-1}, as illustrated in Figure 2.1(c). In practice, a predictive function f_δ is applied for the motion feature prediction at time step $t+\delta$: $\hat{P}_{t+\delta}^{l-1} = f_\delta([P_t^l, P_t^{l-1}])$, where $[\cdot,\cdot]$ denotes channel-wise concatenation. We use a multi-layer perception with one hidden layer for the prediction function: $f_\delta(x) = W_\delta^{(2)}\sigma(W^{(1)}x)$, where σ is ReLU and $W^{(1)}$ is shared across all prediction steps for leveraging their common information.

We define the objective function of each level as a contrastive loss that encourages the predicted $\hat{P}^{l-1}$ to be close to the ground truth P^{l-1} while being far away from the

negative samples:

$$\mathcal{L}^{l}_{\text{contrastive}} = -\sum_{i\in\mathcal{S}}\left[\log\frac{\exp(\text{sim}(\hat{P}_i^{l-1}, P_i^{l-1})/\tau)}{\sum_{j\in\mathcal{S}}\exp(\text{sim}(\hat{P}_i^{l-1}, P_j^{l-1})/\tau)}\right], \tag{2.2}$$

where the similarity function is defined as the cosine similarity as the one used in computing cost volumes, and $\mathcal{S}$ denotes the sampling space of positive and negative samples.

As shown in Figure 2.1(c), the positive sample of the predicted feature is the ground-truth feature that corresponds to the same video and locates at the same position in both space and time as the predicted one. Similar to [29], we define three types of negative samples for all prediction and ground-truth pairs:

- **Spatial negatives**: the ground-truth features that come from the same video of the predicted one but at a different spatial position. Considering efficiency, we randomly sample N spatial locations for each video within a mini-batch to compute the loss. The number of spatial negatives is $(N-1)T^l$.

- **Temporal negatives**: the ground-truth features that come from the same video and locate at the same spatial position, but different time steps. The number of temporal negatives is T^l-1 and they are usually the hard negative samples for self-supervised learning.

- **Easy negatives**: the ground-truth features that come from different videos. The number of easy negatives is $(B-1)NT^l$, where B is the batch size.

As illustrated in Figure 2.1(b), the contrastive motion learning is performed for multiple levels until the motion hierarchy of the whole network is built up.

Progressive Training. Training the multi-level self-supervised learning framework simultaneously from the beginning is infeasible, as the lower-level motion features are initially not well-learned and the higher-level prediction would be arbitrary. To facilitate the optimization process, we propose a progressive training strategy that learns motion features for one level at a time, propagating from low-level to high-level. In practice, after the convergence of training at level $l-1$, we freeze all network parameters up to level $l-1$ (therefore fixing the motion features $\mathcal{P}^{l-1}$), and then start the training for level l. In this way, the higher-level motion features can be stably trained with the well-learned lower-level ones.

2.3.3 Preliminary Motion Cues

To initialize the progressive training, the preliminary motion cues, i.e., $\mathcal{P}^0$, are required as a bootstrap. They should encode some low-level but valid movement information to facilitate the following motion learning. Therefore, we adopt video frame reconstruction to guide the extraction of preliminary motion cues. This task can be formulated as a self-supervised optical flow estimation problem [30, 31], aiming to produce optical flow to allow frame reconstruction from neighboring frames. Motivated by the success of recent work on estimating optical flow with CNNs [62], we build a simple optical flow estimation module using 5 convolutional layers that are stacked sequentially with dense connections. We make use of the optical flow output to warp video frames through bilinear interpolation. The loss function consists of a photometric term that measures the error between the warped frame and the target frame, and a smoothness

term that addresses the aperture problem that causes ambiguity in motion estimation: $\mathcal{L}_{\text{reconstruct}} = \mathcal{L}_{\text{photometric}} + \zeta\mathcal{L}_{\text{smoothness}}$. We define the photometric error as:

$$\mathcal{L}_{\text{photometric}} = \frac{1}{HWT}\sum_{t=1}^{T}\sum_{x=1}^{W}\sum_{y=1}^{H}\mathbb{1}\rho\left(I_t(x,y) - \hat{I}_t(x,y)\right), \tag{2.3}$$

where $\hat{I}_t$ indicates the warped frame at time t and $\rho(z) = (z^2 + \epsilon^2)^\alpha$ is the generalized Charbonnier penalty function with $\alpha = 0.45$ and $\epsilon = 1e^{-3}$ [67], and we apply an indicator function $\mathbb{1} \in \{0, 1\}$ to exclude those invalid positions of the warped pixels that are out-of-boundary. Additionally, we compute the smoothness term as:

$$\mathcal{L}_{\text{smoothness}} = \frac{1}{T}\sum_{t=1}^{T}\rho(\nabla_x U_t) + \rho(\nabla_y U_t) + \rho(\nabla_x V_t) + \rho(\nabla_y V_t), \tag{2.4}$$

where $\nabla_x U/V$ and $\nabla_y U/V$ denote the gradients of estimated flow fields U/V in x/y directions.

2.3.4 Joint Training for Action Recognition

Our ultimate goal is to improve video action recognition with the learned hierarchical motion features. To integrate the learned motion features into a backbone network, we wrap our prime motion block into a residual block: $\mathcal{Z}^l = \mathcal{F}^l + g^l(\mathcal{P}^l)$, where $\mathcal{F}^l$ is the convolutional features at level l, $\mathcal{P}^l$ is the corresponding motion features obtained in Sec. 2.3.2, and $g^l(\cdot)$ is a $1 \times 1 \times 1$ convolution. This seamless integration enables end-to-end fusion of appearance and motion information over multiple levels throughout a single unified network, instead of learning them disjointly like two-stream networks [4]. After the motion representations are self-supervised learned at all levels, we add in the classification loss to jointly optimize the total objective, which is a weighted sum of the

following losses:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{classification}} + \lambda \mathcal{L}_{\text{reconstruct}} + \sum_{l} \gamma^{l} \mathcal{L}^{l}_{\text{contrastive}}, \tag{2.5}$$

where λ and γ^{l} are the weights to balance related loss terms. As shown in Figure 2.1(b), our multi-level self-supervised learning is performed via a side network branch, which can be flexibly embedded into standard CNNs. Furthermore, this self-supervised learning side branch is discarded after training so that our final network can well maintain the efficiency at runtime for inference.

2.4 Experiments

In this section, we first describe in details our experimental setup. We then present a variety of ablation studies to understand the contributions of each individual component in our design, and provide in-depth analysis with qualitative visualization results. In the end, we report comparisons to the state-of-the-art methods on four benchmark datasets.

2.4.1 Experimental Setup

Datasets. We extensively evaluate our proposed approach on the four benchmarks: Kinetics-400 [9], Something-Something (V1&V2) [68] and UCF-101 [69]. **Kinetics-400** is a large-scale video dataset with 400 action categories. As some videos are deleted by their owners over time, our experiments are conducted on a subset of the original dataset with approximately 238K training videos ($\sim$96%) and 196K validation videos ($\sim$98%). In practice, we notice a bit accuracy drop for the same model using our collected dataset with fewer training samples, suggesting that our results can be further improved with

the full original dataset. **Something-Something (V1&V2)** are more sensitive to temporal motion modeling. Something-V1 contains about 100K videos covering 174 classes, and Something-V2 increases videos to 221K and improves video resolution and annotation quality. **UCF-101** includes about 13K videos with 101 classes. As the number of training videos is small, it is often used for evaluating unsupervised representation learning [18, 33] and transfer learning [20, 70].

Implementation Details. Our motion learning module is generic and can be instantiated with various 2D and 3D action networks [9, 10, 71]. In our experiments, we use the standard networks R2D [10] and R(2+1)D [71] as our backbones. We make a few changes to the backbones to improve efficiency, e.g., using bottleneck layers, dropping the temporal convolutions in $\texttt{res}_2$ and $\texttt{res}_3$, and starting temporal striding from $\texttt{res}_3$, as demonstrated in Table 2.1. For the prime motion block, we set $d = 6$ and $s = 2$, and the channel reduction ratio $\beta = 8$. We set the temperature parameter τ in contrastive loss to 0.07 following the practice in previous work [65, 72].

We follow the standard recipe in [12,73] for model training. Specifically, the spatial size of the input is 224×224, randomly cropped and horizontally flipped from a scaled video with the shorter side randomly sampled in [256, 320] pixels. We also apply temporal jittering when sampling clips from a video. We use synchronized SGD training with a cosine learning rate scheduling [74] and a linear warm-up [73]. For the self-supervised training phase, global batch normalization [63] is used to avoid trivial solution [28]. Note that all models are trained from scratch or self-supervised pre-trained without any additional video annotations or pre-computed optical flow.

Layer	R2D-50 / R(2+1)D-101	Output Size
input	–	T × 224 × 224
$\texttt{conv}_1$	1×7×7, 64 stride: 1×2×2	T × 112 × 112
$\texttt{res}_2$	$\begin{bmatrix} 1\times1\times1, 64 \\ 1\times3\times3, 64 \\ 1\times1\times1, 256 \end{bmatrix} \times 3$	T × 56 × 56
$\texttt{res}_3$	$\begin{bmatrix} 1\times1\times1, 128 \\ 1\times3\times3, 128 \\ 1\times1\times1, 512 \end{bmatrix} \times 4$	T/2 × 28 × 28
$\texttt{res}_4$	$\begin{bmatrix} 1\times1\times1, 256 \\ (3\times1\times1, 256) \\ 1\times3\times3, 256 \\ 1\times1\times1, 1024 \end{bmatrix} \times 6$ / 23	T/4 × 14 × 14
$\texttt{res}_5$	$\begin{bmatrix} 1\times1\times1, 512 \\ (3\times1\times1, 512) \\ 1\times3\times3, 512 \\ 1\times1\times1, 2048 \end{bmatrix} \times 3$	T/4 × 7 × 7

Table 2.1: Architectures of the backbone networks R2D-50 / R(2+1)D-101 used in our experiments.

2.4.2 Ablation Study

Supervision for Motion Learning. We first compare the motion features learned at different levels by different supervisions. As our motion learning is based on the self-supervisions that are not directly related with the final action recognition, we first define a measurement to reflect the efficacy of the learned motion features. Towards this goal, we take the extracted motion feature as input and train a lightweight classifier for action recognition on UCF-101. We define the efficacy score as: Score $= \text{Acc}_{\text{train}}/(\text{Acc}_{\text{train}} - \text{Acc}_{\text{test}})$, where $\text{Acc}_{\text{train}}$ and Acc_{test} indicate the top-1 accuracy on the training and test sets. Intuitively, a higher score implies that the representation is more discriminative (with higher training accuracy) and generalizes better (with a lower performance gap between

Input	Supervision			Score
	action	preliminary	hierarchical	
Level 1	✓			2.3
		✓		**3.1**
			✓	–
Level 2	✓			1.7
		✓		2.5
			✓	**3.0**
Level 3	✓			2.4
		✓		1.7
			✓	**3.0**

Table 2.2: Comparison of efficacy scores of the motion features learned at different levels under different supervisory forms.

training and testing).

Table 2.2 shows the efficacy scores of motion features at different levels with different supervisions, where "action" indicates the supervision by action classification, and "preliminary" and "hierarchical" refer to the supervisions by frame reconstruction and contrastive learning. Levels 1, 2 and 3 correspond to the motion features extracted after $\texttt{res}_2$, $\texttt{res}_3$ and $\texttt{res}_4$ of R2D. We observe that the self-supervision of low-level frame reconstruction is particularly effective at level 1, but its performance degrades dramatically at higher levels due to lower spatial/temporal resolutions and higher abstraction of convolutional features. In contrast, the proposed self-supervision by hierarchical contrastive learning is more stable over different levels and more effective to model motion dynamics. It is also observed that the self-supervision, with correct choices at different levels, consistently outperforms the supervision by action classification, which is consistent with the findings in [18,20,33]. In Figure 2.4, we visualize the estimated optical flow, the by-product of frame reconstruction, at each level and find that more accurate optical flow indeed presents at lower levels.

Methods	PMB	Self-Sup	FLOPs	UCF-101	Something-V1	Kinetics-400
Baseline: R2D			1.00×	66.0 / 86.0	36.1 / 68.1	64.8 / 85.1
Ours: R2D	✓		1.18×	71.6 / 89.7	43.6 / 74.7	65.6 / 85.5
Ours: R2D	✓	✓	1.18×	**79.8 / 94.4**	**44.3 / 75.8**	**67.3 / 86.4**
Baseline: R(2+1)D			1.00×	68.0 / 88.2	48.5 / 78.1	66.8 / 86.6
Ours: R(2+1)D	✓		1.11×	73.4 / 92.1	49.2 / 77.9	67.4 / 86.9
Ours: R(2+1)D	✓	✓	1.11×	**80.7 / 95.6**	**50.4 / 78.9**	**68.3 / 87.4**

Table 2.3: Ablation study on the prime motion block (PMB) and self-supervision (Self-Sup) for action recognition. We report computational cost and top-1 / top-5 accuracy (%) on the three benchmarks. Models are evaluated using a single clip per video to eliminate the impact of test-time augmentation.

Contributions of Individual Components. Here we verify the contributions of proposed components by comparing against the baseline, as shown in Table 2.3. We use one clip per video and the 224×224 center crop for evaluation to eliminate the impact of test-time augmentation. We report both top-1 and top-5 accuracy for each setting.

It is obvious that our approach consistently and significantly improves the action recognition accuracy for both 2D and 3D action networks. Our prime motion block provides complementary motion features at multiple levels, and the self-supervision further enhances the representations to encode semantic dynamics. In particular, for the dataset that heavily depends on temporal information like Something-V1, our approach remarkably improves the performance of baseline R2D by 8.2%. For the dataset that is small-scale and tends to overfit to the appearance information like UCF-101, our method improves model generalization and achieves 13.8% improvement. Moreover, our motion learning module only introduces a small overhead to FLOPs of the backbone network.

We next validate the contribution of our motion learning at each level by incrementally adding the proposed motion feature learning block to the baseline. Figure 2.3

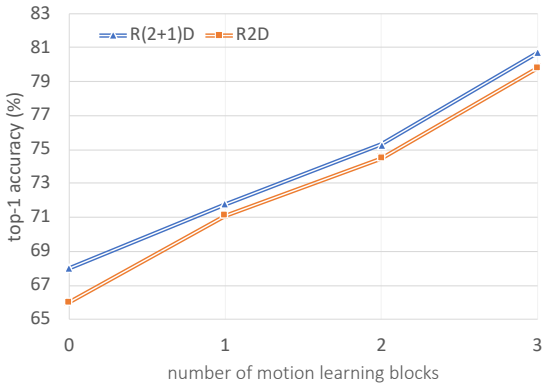


Figure 2.3: Comparison of top-1 accuracy on UCF-101 with incrementally adding the proposed motion learning blocks.

demonstrates the results based on the backbones of R2D and R(2+1)D on UCF-101. We observe that notable gains can be obtained at multiple levels, and the performance gain does not vanish with the increase of motion learning blocks, suggesting the importance of leveraging hierarchical motion information across all levels.

2.4.3 Comparison with State-of-the-Art Results

We compare our approach with the state-of-the-art methods on the four action recognition benchmarks. We report results with the standard test-time augmentations [10]: increasing the spatial resolution to 256×256 and sampling multiple clips per video (9 clips for Something-V1&V2 and 30 clips for Kinetics-400 and UCF-101).

Table 2.4 shows the comparisons on Kinetics-400 and Something-V1&V2. Without using optical flow or supervised pre-training, our model based on backbone R(2+1)D-101 achieves the best results among the single-stream methods over all three datasets. Our approach also outperforms most two-stream methods, apart from the recent two-stream TSM [11] on Something-V2. As for the datasets that focus more on temporal modeling like Something-V1&V2, 2D networks are usually not able to achieve as good results as 3D

Methods	Pre-Training	Flow	Kinetics-400	Something-V1	Something-V2
I3D [9]	ImageNet	✓	75.7	–	–
R(2+1)D [71]	Sports1M		74.3	45.7	–
R(2+1)D [71]	Sports1M	✓	75.4	–	–
NL I3D-50 [10]	ImageNet		76.5	44.4	–
S3D-G [38]	ImageNet	✓	77.2	48.2	–
TRN [75]	ImageNet	✓	–	42.0	55.5
TSM [11]	ImageNet		75.7	49.7	63.4
TSM [11]	ImageNet	✓	–	52.6	**66.0**
ECO [76]	–	✓	70.0	49.5	–
SlowFast [12]	–		77.9	–	–
Disentangling [77]	ImageNet		71.5	–	–
D3D [20]	ImageNet		75.9	–	–
STM [40]	ImageNet		73.7	50.7	64.2
Rep. Flow [41]	–		77.1	–	–
MARS [78]	–		72.7	–	–
DynamoNet [33]	–		77.9	–	–
Ours, R2D-50	–		74.8	46.2	59.4
Ours, R(2+1)D-101	–		**78.3**	**52.8**	64.4

Table 2.4: Comparison of top-1 accuracy (%) with the state-of-the-art methods on Kinetics-400 and Something-V1&V2.

models. However, by equipping with the proposed motion learning module, we find that our method based on backbone R2D-50 outperforms some 3D models, such as R(2+1)D and NL I3D. Our approach also achieves superior results compared with the most recent work that are specifically designed for temporal motion modeling (i.e., the second group in Table 2.4). More importantly, our motion learning is fully self-supervised from raw video frames without any supervisions from optical flow or pre-trained temporal stream.

For the experiment on UCF-101, we fine-tune the models trained only on Kinetics-400 for classification following the standard setting in previous work, and report the averaged accuracy over all 3 splits. As shown in Table 2.5, our approach achieves 97.8% top-1 accuracy, a new state-of-the-art result among the single-stream methods on UCF-101.

Methods	Pre-Training	UCF-101
I3D [9]	ImageNet + Kinetics-400	95.4
R(2+1)D [71]	Kinetics-400	96.8
S3D-G [38]	ImageNet + Kinetics-400	96.8
Disentangling [77]	ImageNet + Kinetics-400	95.9
D3D [20]	ImageNet + Kinetics-400	97.0
STM [40]	ImageNet + Kinetics-400	96.2
MARS [78]	Kinetics-400	97.0
DynamoNet [33]	YouTube8M + Kinetics-400	**97.8**
Ours, R(2+1)D-101	Kinetics-400	**97.8**

Table 2.5: Comparison of top-1 accuracy (%) with the state-of-the-art on UCF-101.

Method	Pre-training	Accuracy
Shuffle and Learn [48]	UCF-101	50.2
OPN [49]	UCF-101	59.8
Odd-One-Out [50]	UCF-101	60.3
ActionFlowNet* [18]	UCF-101	83.9
3D-Puzzel [79]	Kinetics-400	65.8
DPC [29]	Kinetics-400	75.7
DynamoNet [33]	YouTube8M	88.1
Ours: R2D-50 (random)	–	69.9
Ours: R2D-50	Kinetics-400	82.2
Ours: R(2+1)D-101 (random)	–	70.9
Ours: R(2+1)D-101	Kinetics-400	85.1

Table 2.6: Comparison of top-1 accuracy (%) with self-supervised methods on UCF-101 (split-1).

2.4.4 Self-Supervised Pre-Training

The self-supervised learning of video feature representations without using large amount of labeled data has been gaining increasing attention in recent years [29, 33, 48–50, 79]. In addition to the hierarchical motion learning, our multi-level self-supervised learning can also serve as pre-training of a network. As an example, after self-supervised pre-training on Kinetics-400, we fine-tune the network on UCF-101 for 80 epochs with a

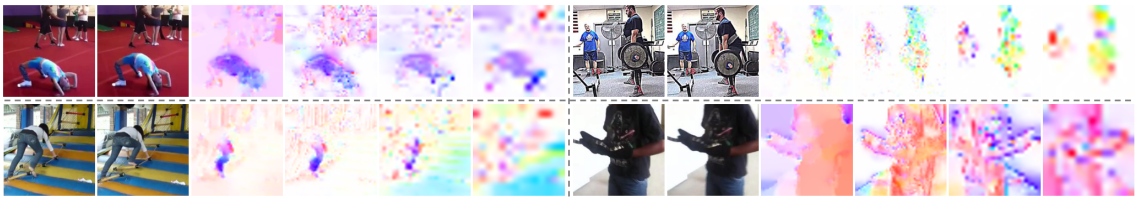


Figure 2.4: Visualization of the estimated optical flow at different feature abstraction levels. For each group, columns 1-2 are adjacent frames; column 3 is the reference optical flow extracted by [80]; columns 4-6 are the estimated optical flow at levels 1, 2 and 3.

batch size of 16 and a learning rate of 0.01. We compare our approach with other state-of-the-art self-supervised methods in Table 2.4.3. Interestingly, even though not specifically designed for network pre-training, our approach is capable of learning effective video representation that generalizes well to the small dataset. Note that previous work requires optical flow [18] or a much larger pre-training dataset (YouTube8M) [33] to achieve state-of-the-art results.

2.4.5 Motion at Different Levels

In this section, we investigate the different semantics integrated in the motion features learned at different levels. This can be justified by the following observations. First, as shown in figure 2.4, the motion features at lower levels generate more accurate optical flow, indicating that they involve more pixel-wise and low-level motion information than the ones at higher levels.

Second, we conduct a video retrieval experiment using the learned motion features of different levels. Specifically, given a query video, we rank the videos in the validation set based on the ℓ_2 distance of their motion features extracted at a certain level. As illustrated in Figure 2.5, the motion features at lower levels tend to retrieve the videos sharing similar elementary movements but lacking of high-level action correlations. Fur-

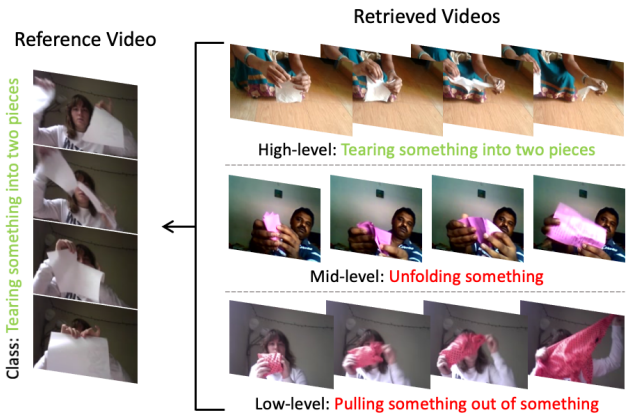
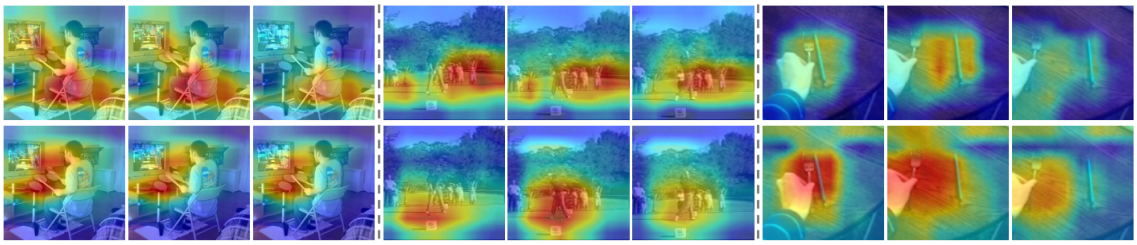


Figure 2.5: Examples of the retrieved videos to reflect the different motion semantics learned by the motion features at different levels.

	Acc (%)	Classes with the largest relative gains
Low-level	11.6	"turning the camera left" (73.8) "turning the camera right" (71.7) "turning the camera upwards" (58.3)
Mid-level	15.2	"showing a photo to the camera" ($0 \rightarrow 19.4$) "showing smth behind smth" ($0 \rightarrow 12.2$) "poking a stack of smth" ($0 \rightarrow 9.5$)
High-level	21.7	"pulling two ends of smth" ($0 \rightarrow 14.0$) "tipping smth with smth in it over" ($0 \rightarrow 12.5$) "sprinkling smth onto smth" ($0 \rightarrow 9.1$)

Table 2.7: Comparison of classification accuracy using motion features learned at different levels (accuracy for random output: 0.6%). For the mid/high levels, we show the top-3 classes with the largest relative gains compared with the lower-level motion features. For the low level, we report the top-3 classes with the highest accuracy instead.

thermore, the retrieval results can be used as a k-nearest-neighbor classifier by assigning the query video to the majority class among its top k retrieval results. We compare the classification accuracy for motion features learned at different levels in Table 2.7. It is not surprising that the motion features at higher levels achieve higher accuracy than the ones at lower levels as the former possess more useful semantics of motion dynamics for the action recognition task. More interestingly, we find that the low-level motion features can obtain relatively high accuracy for some action classes with apparent moving patterns



(a) Baseline: PlayingCello Ours: Drumming (b) Baseline: CricketBowling Ours: GolfSwing (c) Baseline: Moving sth closer to something Ours: Moving sth away from something

Figure 2.6: Visualization of the learned features by Grad-CAM++ [81]. Fonts in green and red indicate correct recognition and misclassification. (a-b) Features learned by our approach (bottom) are more sensitive to the regions with motion cues. (c) Our motion learning module (bottom) equips the 2D backbone with the ability of reasoning the temporal order of video frames.

(e.g., "turning the camera left"). This indicates that the low-level motion features are capable of extracting elementary movements from raw video frames. On the other hand, the motion features at higher levels can recognize the actions that require finer understanding of high-level motion semantics (e.g., "pulling two ends of something so that it separates into two pieces").

2.4.6 Qualitative Results

To qualitatively verify the impact of the learned motion features, we utilize Grad-CAM++ [81] to visualize the class activation map of the last convolution layer. Figure 2.6 shows the comparison between baseline and our model with the backbone R2D-50 on UCF-101 and Something-V1. Our model attentions more on the regions with informative motion, while the baseline tends to be distracted by the static appearance. For instance, in Figure 2.6(a), our method focuses on the moving hands of the person, while the baseline concentrates on the static human body. Our motion learning module also equips the 2D backbone with effective temporal modeling ability. As shown in Figure 2.6(c), our

model is capable of reasoning the temporal order of the video and predicting the correct action, while the baseline outputs the opposite prediction result as it fails to capture the chronological relationship.

2.5 Conclusion

We have presented hierarchical contrastive motion learning, a multi-level self-supervised framework that progressively learns a hierarchy of motion features from raw video frames. A discriminative contrastive loss at each level provides explicit self-supervision for motion learning. This hierarchical design bridges the semantic gap between low-level motion cues and high-level recognition tasks, meanwhile promotes effective fusion of appearance and motion information to finally boost action recognition. Extensive experiments on four benchmarks show that our approach compares favorably against the state-of-the-art methods yet without requiring optical flow or supervised pre-training.

Chapter 3: Global Temporal Attention for Video Action Understanding

3.1 Introduction

Attention mechanisms have demonstrated impressive achievements in a wide range of tasks such as language modeling [82, 83], speech recognition [84] and image classification [85, 86]. One of the most effective attention methods is self-attention, which learns self-alignment via dot product operations, computing pairwise similarities between a pixel (*i.e.*, query) and other pixels (*i.e.*, key) to modulate the transformed inputs (*i.e.*, value). For action recognition [10], this requires: (i) flattening all pixels in a video, regardless of their spatial and temporal locations, into a huge vector; (ii) sharing the same set of parameters for all pixels to derive the query/key/value; and (iii) generating a joint attention map for both spatial and temporal context.

In this chapter, we seek a better understanding of self-attention for temporal modeling in videos. In particular, we wish to answer the following questions: (i) Is treating all pixels in space and time as a flattened vector to perform dot-product sufficient for temporal modeling? (ii) Is dot product based self-attention really necessary for capturing temporal relationships across different frames?

In contrast to the conventional use of self-attention for video recognition, we posit that temporal attention should be *disentangled* from spatial attention, since they focus on

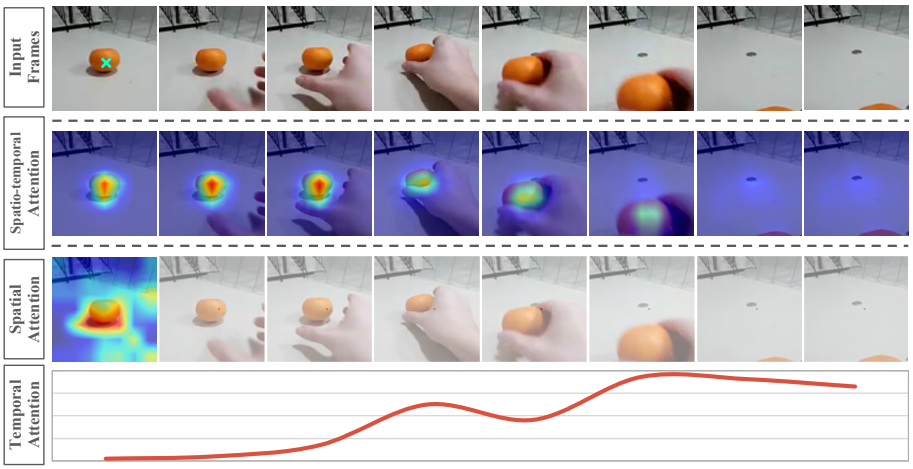


Figure 3.1: **Top**: input frames (action: *removing something to reveal something*). The green cross-mark indicates the query position. **Center**: spatio-temporal attention maps generated by NL blocks. The attention is biased towards the appearance similarity, which fades overtime ignoring temporal clues; thus, the model generates incorrect prediction: *putting something in front of something*. **Bottom**: the decoupled blocks generate spatial attention maps within the query frames and temporal attention weights across different time steps. The temporal attention has larger values at the key frames that are critical for recognizing the action (*i.e.*, revealing something), and the model gives the correct prediction. GTA is built upon the decoupled framework and advances the temporal attention to a more effective design.

different aspects. As shown in Figure 3.1, the spatial attention tends to capture appearance similarity (*i.e.*, the orange), while the temporal attention is more focused on frames that are important for recognizing the action (*i.e.*, revealing something). When these two types of attention are modeled together (Figure 3.1 Center), the attention is biased towards the appearance similarity, dominating any temporal context.

In addition, we argue that dot product based self-attention is *not even suitable* for temporal modeling. Standard self-attention produces instance-specific attention weights, conditioned on pairwise interactions. In the spatial domain, it can attend to salient regions for improved performance. When used for temporal modeling, it ignores the ordering of frames as self-attention is known to be permutation invariant [87]. For instance, if we shuffle two pixels temporally, their relationship will be the same as before, producing the

same output. This is not sufficient for differentiating actions like "reveal something" and "cover something". We hypothesize that temporal modeling requires learning a global temporal structure that generalizes across different samples rather than relying on pairwise interactions across time steps.

In light of this, we introduce Global Temporal Attention (GTA), for video action recognition. In particular, we first decouple the traditional spatio-temporal self-attention into two successive steps—a standard self-attention in the spatial domain within each frame followed by the proposed GTA module to capture temporal relationships across different frames. Moreover, we not only apply GTA to each pixel location along the temporal dimension but also "superpixels"—pixels in a region share similar semantic meanings. This enables our model to capture temporal relationships at different levels of spatial granularity. Unlike computing pairwise frame interactions with dot product, GTA directly learns a global attention matrix that is randomly initialized to be instance-independent. The intuition of the global attention matrix is to not rely on pairwise frame relations without specific ordering information or individual sample information, but to learn a global task-specific weight matrix considering temporal structures that generalize across different samples. To exploit information across different channels, we split feature maps into multiple groups along the channel-dimension, and for each group we apply GTA in a multi-head fashion such that each head focuses on different aspects of the inputs. Then, outputs from different channel groups are further aggregated to produce a unified representation.

We conduct extensive experiments on Something-Something [68] (both v1 and v2) and Kinetics-400 [88]. Our experimental results demonstrate that our proposed GTA

outperforms the traditional spatio-temporal self-attention by clear margins, and achieves state-of-the-art results on these three datasets. We also provide a side-by-side comparison with recent NL variants [89–91] to show the superior performance of GTA in temporal modeling. We summarize our main contributions as follows.

- We provide an in-depth analysis of the sub-optimal design of the spatio-temporal self-attention and propose to decouple attention across the two dimensions.
- We introduce GTA, which improves the conventional temporal attention by introducing: (i) temporal modeling at both pixel and region levels; (ii) a global attention matrix for all samples; (iii) a cross-channel multi-head design for incorporating channel interactions.

3.2 Related Work

Temporal Modeling in Action Recognition. A large family of research in action recognition focuses on the effective modeling of temporal information in videos. Early work simply aggregates the frame/clip-level features across time via average pooling [7, 92] or feature encoding like ActionVLAD [8], without considering the temporal relationships of video frames. Later on, two-stream networks [4], 3D convolution networks (CNNs) [6,93] and recurrent neural networks (RNNs) [5,94] are used to model the spatial and temporal context in videos. Recently, various temporal modules are proposed to capture temporal relations, such as TRN [75] based on relation networks, Timeception [95] based on multi-scale temporal convolutions, and SlowFast [12] based on slow and fast branches capturing spatial and motion information, respectively. TSM [11] adopts a chan-

nel shifting operation along the time dimension to enable temporal modeling on 2D CNN networks. STM [40], TEA [96] and MSNet [44] encode the motion information into the network by extracting motion features between adjacent frames.

Non-Local and Self-Attention. Modeling long-range relations in feature representations has a long history [97–102] and has proven to be effective in various tasks, such as machine translation [83], generative modeling [103], image recognition [10, 86, 90], object detection [10, 90, 104], semantic segmentation [10, 90, 105] and visual question answering [106]. In computer vision, Non-local Network (NL) [10] is proposed to model the pixel-level pairwise similarities to encode long-range dependencies. SENet [85] uses a Squeeze-and-Excitation block to model inter-dependencies along the channel dimension. GCNet [90], CGNL [89] and DANet [107] further improve the vanilla NL by integrating pixel-wise and channel-wise attention. CCNet [108] improves the efficiency of NL by computing the contextual information of the pixels on its crisscross path instead of the global region. GloRe [91] proposes the relation reasoning via graph convolution on a region-based graph in the interaction space to capture the global information.

In this work, we present a novel way to model temporal relationships and bring new perspectives for a better understanding of the attention mechanism utilized in video action recognition. Our approach learns global temporal attention that generalizes well across different samples as opposed to using pairwise interactions with dot product in the conventional self-attention.

3.3 Approach

We start with a brief review of self-attention for video action recognition in Section 3.3.1. Then, we introduce the decoupling of spatial and temporal attention in Section 3.3.2. Finally, we elaborate on GTA that is designed for better temporal modeling in Section 3.3.3 and 3.3.4.

3.3.1 Background

Extending the self-attention module [83] for language tasks, the non-local block [10] takes as input flattened pixels in spacetime to model pairwise interactions, as shown in Figure 3.2(a). More formally, given an input feature map $X \in \mathbb{R}^{N \times C}$, three linear projections are applied to obtain key (K), query (Q), and value (V) representations, where C is the channel dimension of the feature map. We use $N = THW$ to denote the total number of positions in both space and time dimensions, where T, H and W are the number of time steps, height and width of the feature map, respectively. The three projections can be written as

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V, \tag{3.1}$$

parameterized by three weight matrices $W_Q, W_K, W_V \in \mathbb{R}^{C \times C}$ respectively. The output of the self-attention operation is computed as a weighted sum of the value representations. Here, the weight is defined by the attention weight matrix $M \in \mathbb{R}^{N \times N}$, where each element denotes a scaled dot product between the query pixel and the corresponding key

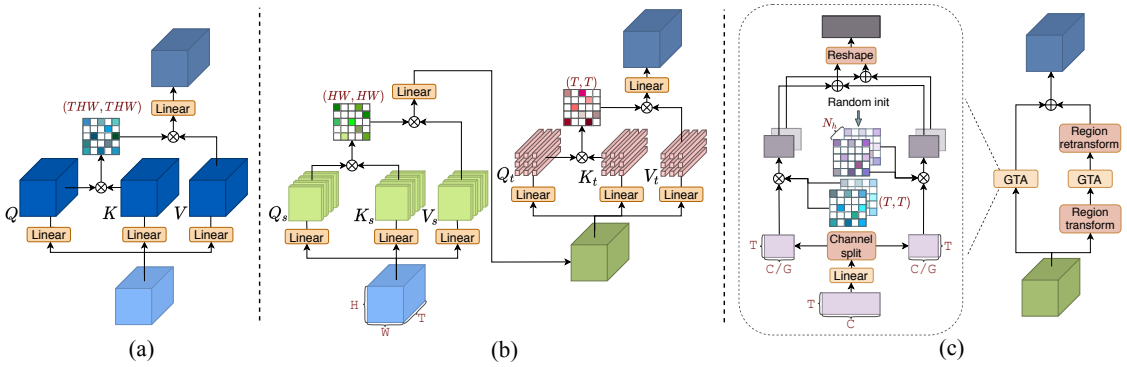


Figure 3.2: (a) **Standard self-attention for action recognition**, which computes pairwise similarities between a pixel (query) with other pixels (key) in the spacetime domain. (b) **Decoupled spatial and temporal self-attention**, which uses separated key/query/value representations for spatial and temporal attention and aggregates spatial and temporal context in a separate manner. (c) **Global temporal attention**, which learns two randomly initialized global attention maps at the pixel-level and the region-level, respectively. Regions are derived automatically with a learned transformation matrix. Inside the rectangular (dashed line), the spatial dimension of feature maps is omitted. GTA is also applied in a cross-channel multi-head fashion, where feature maps are split along the channel dimension into G groups (only 2 groups are shown for simplicity). See texts for more details.

pixel, followed by a softmax normalization:

$$M = \texttt{softmax}\left(\frac{QK^T}{\sqrt{C}}\right) \tag{3.2}$$

$$A = MV \tag{3.3}$$

The attention output is incorporated into the backbone network via a final linear projection and a residual connection [34]:

$$Y = X + AW_O, \tag{3.4}$$

where $W_O \in \mathbb{R}^{C\times C}$. An optional normalization layer (*e.g.*, BatchNorm [63] and LayerNorm [109]) can be used before the residual connection, and we drop it here for clarity.

3.3.2 Decoupled Spatial and Temporal Self-Attention

Although self-attention has been widely used in action recognition for capturing spatio-temporal dependencies, we argue in this chapter that the coupled modeling of spatial and temporal self-attention prevents the model from learning effective temporal attention. First, when sharing the same transformation matrices for key, query and value, it fails to differentiate between spatial and temporal contexts. This is unsatisfactory for temporal modeling as we need to consider temporal structures of videos instead of simply computing the salient regions by performing self-attention in the spatial domain. Moreover, when the two attentions are modeled and aggregated jointly, the combined attention tends to be biased towards the appearance similarity as the temporal attention is dominated by the spatial one (see Figure 3.1). Based on this observation, we propose the decoupled spatial and temporal self-attention in Figure 3.2 (b), which breaks down the standard self-attention block into a spatial self-attention block followed by a temporal self-attention block. We will provide a more in-depth analysis of the decoupled self-attention design via experiments in Section 3.4.2. Note that although the idea of processing spatial and temporal information separately has been explored before for video understanding [71, 110], the effect of disentangling the two dimensions in self-attention is *unknown* in prior work.

3.3.3 Global Temporal Attention

We now introduce GTA, which is built upon the decoupled self-attention framework and advances the temporal attention to a more effective design. GTA aims to learn a global attention map that considers temporal structures and generalizes well for all samples.

Formally, given the input feature map $X \in \mathbb{R}^{T\times HW\times C}$ generated by the spatial self-attention block, GTA models temporal relationships at two different levels of spatial granularity: *pixel-level* and *region-level*. For Pixel GTA, all positions in the spatial domain (*i.e.*, HW) are treated individually as different samples and temporal modeling is performed along the time axis T. As for Region GTA, we first project the spatial domain HW of each time step to R semantic regions. This is achieved by a region transformation matrix $W_R \in \mathbb{R}^{T\times R\times HW}$ that groups similar pixels with related semantic meanings into the same region [91]. Then, temporal modeling is performed across frames on each region individually in the same manner as Pixel GTA, followed by a reversed region transformation matrix W_R^T to reproduce the pixel-level spatial domain. Similar to Eqn. 3.4, the output of GTA can be written as:

$$Y = X + \underbrace{A^P W_O^P}_{\text{Pixel GTA}} + \underbrace{W_R^T A^R W_O^R}_{\text{Region GTA}}, \tag{3.5}$$

where W_O^P and W_O^R transform the output of Pixel GTA and Region GTA to be added with X in a residual manner.

Unlike conventional self-attention where the attention map is produced by pairwise interactions conditioned on input pixels (Eqn. 3.2), we train an attention map that does *not* depend on individual pixels. In particular, we directly learn a randomly initialized weight matrix $\hat{M} \in \mathbb{R}^{T\times T}$ to modulate the value representation:

$$A = \hat{M}V. \tag{3.6}$$

We omit the superscript P and R for A, $\hat{M}$ and V, as the same operations are applied to both Pixel GTA and Region GTA. The idea of using a global attention matrix rather than pairwise dot product is that pairwise interactions fluctuate across different samples, lacking a global temporal consistency at the dataset level. In addition, the standard self-attention fails to consider the ordering of sequences [87]—if we shuffle the pixels used to compute the attention map (*i.e*. Eqn. 3.2), the attention value between a pair would still be the same in the matrix, thus the output will not change, which is not what we desire.

Cross-channel Multi-head GTA. The attention matrix $\hat{M}$ in Eqn. 3.6 is used to learn a linear combination of $V \in \mathbb{R}^{T\times C}$ [1] across different time steps, without considering feature interactions in the channel dimension. We further improve temporal modeling by incorporating channel interactions. We split C into G groups, and for each group and we apply a multi-head GTA. In particular, for the g-th group, the outputs of the multi-head attention $\texttt{MH}_g$ can be derived as:

$$\texttt{MH}_g = \texttt{Concat}_{k=1}^{N_h}(\hat{M}_g^k V_g) \in \mathbb{R}^{N_h\times T\times \lfloor \frac{C}{G} \rfloor}, \tag{3.7}$$

where $\hat{M}_g^k \in \mathbb{R}^{T\times T}$ represents the k-th head for the g-th group, $V_g \in \mathbb{R}^{T\times \lfloor \frac{C}{G} \rfloor}$ denotes the value for the g-th group and N_h denotes the number of heads used. Each head focuses on distinct temporal attention patterns. To capture interactions across different groups, we sum the outputs along the channel dimension between different groups to produce $\texttt{MH}_G$

[1] HW are considered as different samples and we omit it for brevity.

as:

$$\texttt{MH}_G = \sum_{g=1}^{G} \texttt{MH}_g \in \mathbb{R}^{N_h \times T \times \lfloor \frac{C}{G} \rfloor}, \tag{3.8}$$

which mixes information across channels in different groups. In order for $\texttt{MH}_G$ to have the same size as $X \in \mathbb{R}^{T \times C}$ [1] for residual addition, one can transform $\texttt{MH}_G$ with an additional layer. Instead, we simply set N_h to be G and reshape $\texttt{MH}_G$ to be the same size of $\mathbb{R}^{T \times C}$.

3.3.4 Relations to Prior Work

Our proposed decoupled framework and the cross-channel multi-head (CCMH) design are the two key differences between GTA and the prior work (GloRe [91]). Specifically, our Region GTA generates semantic regions within each frame and performs temporal modeling on each region *individually* along the time axis. In contrast, when applied to spatio-temporal data, GloRe projects the whole 3D feature maps into semantic groups and models the interactions among them. We argue that this kind of grouping and modeling is not capable of capturing effective temporal relationships across different time steps. Moreover, GloRe leverages graph convolution to model node-wise interactions, which only considers information diffusion on each channel. Our GTA incorporates channel interactions to further improve temporal modeling, and we show its effectiveness in Section 3.4.3.

3.4 Experiments

3.4.1 Experimental Setups

Datasets. We evaluate our approach on three video action benchmarks, including two temporal-related datasets: Something-Something v1 (SSv1) and Something-Something v2 (SSv2) [68], and a large-scale dataset that is less sensitive to temporal relationships: Kinetics-400 (K400) [9]. In detail, Something-Something v1&v2 include 110K and 220K videos respectively over 174 action classes. Kinetics-400 is a human action recognition dataset that consists of about 300K YouTube videos covering 400 classes. As stated in [38,75,111], actions in Kinetics datasets can be easily determined from the background clues instead of temporal information. As we aim to improve temporal modeling for video action recognition, our experiments focus more on temporal sensitive datasets (SSv1 and SSv2).

Implementation Details. GTA is flexible and can be easily inserted into existing 2D and 3D backbones. In our experiments, we adopt the standard R2D-50 network [34] and the SlowFast-R50 network [12] as our 2D/3D backbones.

For experiments using 2D CNN backbones, we initialize the model with ImageNet [112] pre-trained weights and follow the segment-based sampling strategy [7] for model training. We also apply the same data augmentation as TSN [7], which first resizes the input frames to 240 $\times$ 320 pixels, followed by the multi-scale cropping and random flipping, and then resizes the cropped regions to 224 $\times$ 224 pixels. Unless otherwise stated, we use 8 frames as inputs for experiments on 2D CNN backbones for SSv1 and

SSv2 datasets. For experiments with 3D CNN backbones, we follow the same sampling practice and training/testing strategy in SlowFast [12]. Specifically, we densely sample a clip from a full video and apply scale-jittering with a shorter side sampled in [256,340] pixels and then randomly crop a region of 224 $\times$ 224 pixels for training. More dataset-specific training details are available in the supplementary material.

During testing, on SSv1 and SSv2 datasets, we sample 1 clip from each video and the center crop of size 224$\times$ 224 pixels for evaluation. We keep the same protocol for the methods compared in the same table. On the Kinetics-400 dataset, we sample 10 clips in the temporal domain and 3 crops in the spatial domain of size 256$\times$ 256 pixels.

3.4.2 Main Results

Effectiveness of GTA in a decoupled framework. We report the results of GTA using both 2D and 3D backbones and compare with the alternative approaches: (1) standard non-local block (NL) [10], which is a variant of self-attention that flattens all pixels in space and time dimension into a huge vector; (2) decoupled non-local block (DNL), which breaks down the non-local block into spatial self-attention followed by temporal self-attention. For both of our approaches and the compared baselines, we apply five blocks (2 to res_3 and 3 to res_4 for every other residual block) in the backbone networks unless specified, following [10].

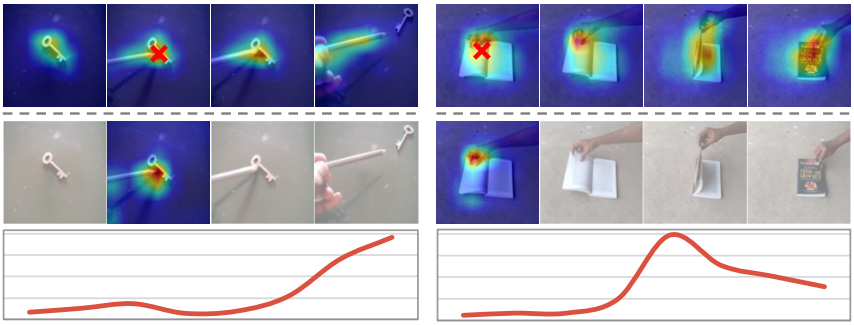
Table 3.1 summarizes the comparison results. We first observe a huge gap between the performance of 2D and 3D backbones, which shows the importance of utilizing temporal information for SSv1&SSv2 datasets. Notably, we see that by simply separating

Model	FLOPs (G)	#Params (M)	SSv1 (%)	SSv2 (%)
R2D-50	32.7	23.9	17.0	26.8
+ NL	61.1	31.2	31.2	50.7
+ DNL	49.9	31.2	38.8	55.5
+ GTA	50.2	31.2	**50.6**	**63.5**
SlowFast-R50	131.4	34.0	50.9	63.4
+ NL	239.9	41.4	51.7	63.9
+ DNL	169.1	41.4	52.0	64.1
+ GTA	169.9	41.4	**53.4**	**64.9**

Table 3.1: Comparisons with the standard non-local block (NL). Top-1 accuracy on validation set is reported here. GTA significantly outperforms NL on both 2D/3D backbones while requiring 20%-30% *less* computation cost.

temporal self-attention from spatial self-attention, DNL outperforms NL on both backbones, while requiring 20%-30% less computation cost. Compared to NL, DNL offers a 7.6% / 4.8% gain on SSv1 and SSv2, respectively in the 2D setting. This suggests that the spatial and temporal self-attentions should be treated *separately* to capture more informative temporal contexts.

We further visualize the attention maps obtained by NL and DNL in Figure 3.3. The first row in each example shows the attention maps obtained by NL, which includes spatio-temporal attention between the query position and all other positions in the entire video. The second and third rows show the attention maps obtained by DNL, including a spatial attention map within the query frame and a temporal attention map at the query position over the whole sequence. We observe that the attention maps obtained by the NL mostly focus on the same object across input frames (*i.e.*, the key, the book) and are uniformly distributed across different time steps. This implies that the attention in NL mostly concentrates on capturing the appearance similarities instead of the temporal



(a) (b)

Figure 3.3: Visualization of the attention maps of two examples: (a) *pushing something with something*; (b) *closing something*. The first row shows the spatio-temporal attention maps generated by the non-local block. The red cross-mark denotes the query position. The second and third rows are the spatial and temporal attention maps obtained by the decoupled non-local block.

relations within the video sequence. In contrast, by decoupling them, DNL is capable of capturing effective temporal dependencies by attending to the frames which are critical for recognizing the actions. For example, in Figure 3.3(a), the temporal attention weights are larger in the last few frames where the action "pushing something" is happening, while the temporal attention weights in the middle frames are greater in Figure 3.3(b).

Finally, GTA produces the best results on the two datasets with both 2D and 3D backbones with reduced FLOPs comparing to NL. For example, on the 2D backbone, GTA further outperforms DNL by 11.8% / 8.0% on SSv1, SSv2, respectively, confirming the effectiveness of GTA for temporal modeling. On a 3D backbone, we observe similar trends with smaller gains, since GTA is built upon a stronger baseline with temporal information extracted by 3D convolutions. This highlights the compatibility of GTA for both 2D and 3D networks. It is also noteworthy that 2D networks can achieve comparable performance with 3D backbones when equipped with GTA.

Setting	CGNL [89]	GCNet [90]	GloRe [91]	GTA
Original	26.7	28.4	33.2	–
Decoupled	37.4	39.0	38.6	**50.6**

Table 3.2: Comparisons with recent NL variants using the R2D-50 backbone. Top-1 accuracy (%) on SSv1 is reported.

Temporal modeling in NL variants. Recent work has focused on improving the vanilla non-local block by introducing channel-wise attention [89, 90] or graph-based reasoning [91]. Although these variants have been applied to video action recognition, their capacity to model temporal relations is relatively underexplored. In Table 3.2, we provide a side-by-side comparison with these NL variants and their decoupled version on SSv1. The decoupled variants are achieved by breaking down the spatio-temporal operations into spatial operations at each frame followed by temporal operations at each pixel.

We first observe that all three variants fail to achieve satisfying improvements over the vanilla NL (31.2%). In particular, the use of extra channel-wise attention (CGNL [89], GCNet [90]) leads to even worse results, indicating that the entangled modeling of spatial, temporal and channel interactions in fact hinders the learning of temporal relationships. Interestingly, by simply decoupling the spatial and temporal operations, substantial improvements can be achieved for all three variants and the results are comparable with DNL (38.8%). Nevertheless, our GTA outperforms these NL variants by clear margins, which demonstrates its superior capacity to model temporal information.

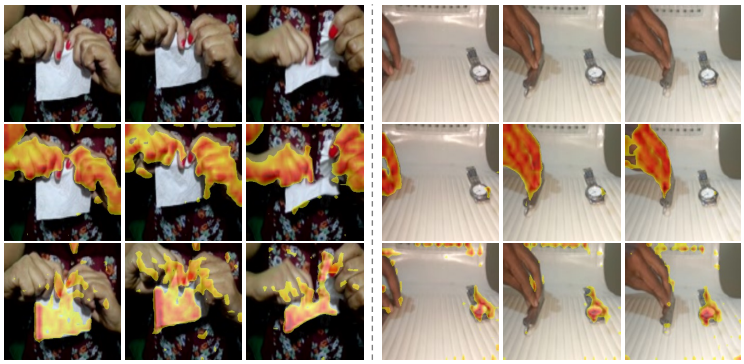
Pixel	Region	CCMH	Top-1	Δ
✓	✓	✓	50.6	–
✓		✓	49.6	-1.0
	✓	✓	47.9	-2.7
✓	✓		49.4	-1.2
✓			49.1	-1.5
	✓		46.3	-4.3

Table 3.3: Ablation on the contribution of different components in GTA. Group count is set to 8 for CCMH.

3.4.3 Ablative Studies

Contribution of Different Components. We first validate the contribution of each component in GTA by removing them from the full model. As shown in Table 3.3, while Pixel GTA plays a more important role than Region GTA, the combination of these two modules yields the best result, achieving more than 1% improvement compared to using each of them alone. It indicates that Pixel GTA and Region GTA are *complementary* to each other, focusing on learning temporal relationships at different levels of spatial granularity. We further visualize regions that are automatically discovered by Region GTA in Figure 3.4. We can see that Region GTA is capable of discovering regions that share similar semantic meanings. For example, in the first video, the "hand" and the "paper" are automatically identified as different regions, while the "hand" and the "watch" are detected in the second video.

Table 3.3 also shows the contribution of the cross-channel multi-head (CCMH) design when the group size is set to 8. Specifically, CCMH has a larger impact on Region GTA than Pixel GTA (1% gain v.s. 0.5% gain) and we hypothesize that modeling temporal



(a) (b)

Figure 3.4: Visualization of transformed regions: (a)“Tearing something into two pieces”; (b)“Moving something closer to something”. The first row is the frame sequences. The second and third rows are regions obtained by Region GTA.

Model	**FLOPs (G)**	**#Params (M)**	**Top-1**	Δ
R2D-50	32.7	23.9	17.0	-
+ SA	41.7	27.5	17.9	+0.9
+ TA	41.0	27.5	37.6	+20.6
+ SA + TA	49.9	31.2	38.8	+21.8
+ SA + TAPE	49.9	31.2	48.4	+31.4
+ SA + GTA	50.2	31.2	50.6	+33.6

Table 3.4: Ablation on the impact of SA and temporal order. GTA achieves superior performance with minor extra FLOPs compared with TAPE.

relationships at the region level is more challenging and requires channel interactions. With the improved performance of Region GTA, the fusion of pixel-level and region-level information becomes more beneficial when CCMH is applied (1.0% gain v.s. 0.3% gain w/o CCMH).

Impact of Spatial Attention and Temporal Order. In Table 3.4, we compare the contribution of spatial and temporal self-attention modules, as well as the impact of modeling temporal order in temporal self-attention. As the SSv1 dataset relies highly on temporal relationships, applying spatial self-attention (SA) alone in the spatial domain slightly im-

res$_3$	**res$_4$**	**res$_5$**	**Top-1**
			17.0
+1			46.2
	+1		46.4
		+1	37.4
+1	+1		49.5
+2	+3		50.6

Table 3.5: Ablation on the impact of inserting positions and number of blocks.

proves the backbone network (0.9% gain). In contrast, using the temporal self-attention (TA) provides much more significant improvements (20.6% gain). Adding positional encoding to the temporal self-attention module (TAPE) further improves the performance by 9.6%, which proves the importance of modeling temporal order information. Finally, our GTA achieves the best result with a negligible increase in computation cost. It is worth noting that our Pixel GTA (without applying GTA to regions) already outperforms TAPE no matter whether CCMH is used or not (49.1% / 49.6% in Table 3.3). This verifies that our GTA design is more effective in temporal modeling than temporal self-attention and positional encoding.

Impact of inserting positions and number of blocks. Table 3.5 explores the performance of different inserting positions and the number of blocks inserted. We see that even a single GTA block inserted at res$_3$ or res$_4$ can bring significant improvement over the baseline. However, the enhancement on res$_5$ is relatively minor. We hypothesize that the final residual stage loses too much fine-grained spatial information, which may hinder the learning of temporal attention at the pixel-level and the region-level. Following the common practice [10], our full model inserts five GTA blocks to leverage the comple-

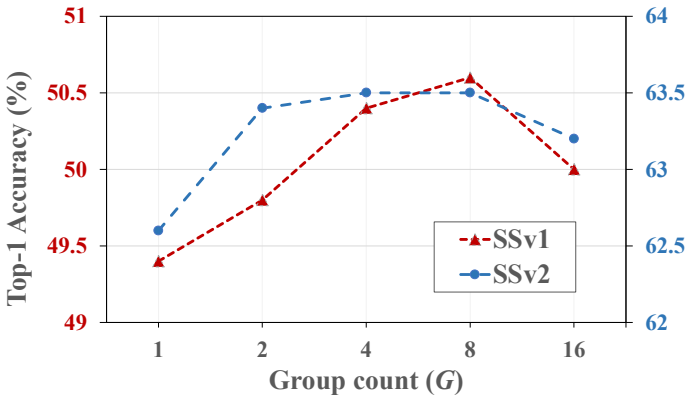


Figure 3.5: Impact of group count in cross-channel multi-head GTA.

Model	w/o CCMH	w/ CCMH
+ TAPE	46.5	47.2
+ Pixel GTA	**48.0**	**48.5**
+ SA + TAPE	48.4	48.8
+ SA + Pixel GTA	**49.1**	**49.6**

Table 3.6: Ablation on positional embedding (TAPE) and cross-channel multi-head (CCMH) design.

mentary information provided by different residual stages and achieves the best result.

Impact of Group Count in CCMH GTA. We evaluate different values of group count used in GTA on SSv1 and SSv2 datasets. The results are summarized in Figure 3.5. We can see that using a group count larger than 1 can largely improve the performance on both SSv1 and SSv2 datasets, which demonstrates the importance of channel interactions in GTA. And a group count of 8 offers the best performance on SSv1 and SSv2. When the group count becomes larger than 8, the performance drops because the number of channels in each group becomes too small.

Comparison with Temporal Attention with Positional Embedding (TAPE) Our GTA module is more effective in temporal modeling than TAPE because it not only consid-

Method	GFLOPs × views	Top-1	Top-5
TSN [7]	80×10	72.5	90.2
TSM [11]	86×30	74.7	91.4
bLVNet-TAM [113]	93×9	73.5	91.2
MSNet [44]	87×10	76.4	-
Two-Stream R(2+1)D [71]	304×115	75.4	91.9
Two-Stream S3D-G [38]	143×N/A	77.2	93.0
I3D+NL [10]	359×30	77.7	93.3
SlowFast-R101 [12]	106×30	77.9	93.2
CorrNet-R101 [19]	224×30	79.2	-
R2D-R50 + NL	77×30	74.8	91.5
R2D-R50 + GTA	62×30	75.9	92.2
SlowFast-R101 + NL	137×30	78.9	93.9
SlowFast-R101 + GTA	137×30	**79.8**	**94.1**

Table 3.7: Comparisons with state-of-the-art methods on Kinetics-400 dataset.

ers the chronological order of video frames but also models the temporal relationships among them. Results in Table 2(b) of the main paper show that GTA outperforms TAPE by **2.2%** on SSv1. Here, we provide a side-by-side comparison between TAPE and our Pixel GTA (without applying GTA to regions) in Table 3.6. Our Pixel GTA consistently outperforms TAPE under different settings. Furthermore, TAPE can also benefit from our cross-channel multi-head (CCMH) design, but Pixel GTA still achieves the best performance.

3.4.4 Comparison with State-of-the-art

Kinetics-400 Table 3.7 presents the comparative results with other state-of-the-art methods on Kinetics-400. The first section of the table shows the methods based on 2D CNN network. The second section contains the models with 3D CNN backbone. The third sec-

Method	Backbone	Frames×Crops ×Clips	SSv1		SSv2	
			Val Top-1	**Val Top-5**	**Val Top-1**	**Val Top-5**
TRN [75]	BNInception	8×1×1	34.4	-	48.8	-
TSM [11]	2D R50	8×1×1	45.6	74.2	58.8	85.4
TSM [11]	2D R50	16×1×1	47.3	77.1	61.2	86.9
$TSM_{RGB+Flow}$ [11]	2D R50	(16+16)×1×1	52.6	81.9	65.0	89.4
MSNet [44]	2D R50+TSM	8×1×1	50.9	80.3	63.0	88.4
MSNet [44]	2D R50+TSM	16×1×1	52.1	82.3	64.7	89.4
$MSNet_{En}$ [44]	2D R50+TSM	(16+8)×1×10	<u>55.1</u>	**84.0**	<u>67.1</u>	<u>91.0</u>
ECO [76]	3D R18+BNInc	16×1×1	41.4	-	-	-
ECO_{En}*Lite* [76]	BNInc+3D R18	92×1×1	46.4	-	-	-
I3D+NL [10]	3D R50	32×3×2	44.4	76.0	-	-
I3D+NL+GCN [114]	3D R50	32×3×2	46.1	76.8	-	-
S3D-G [38]	3D Inception	64×1×1	48.2	78.7	-	-
CorrNet [19]	3D CorrNet-50	32×1×10	48.5	-	-	-
CorrNet [19]	3D CorrNet-101	32×3×10	51.1	-	-	-
TEA [96]	3D R50	8×1×1	48.9	78.1	-	-
TEA [96]	3D R50	16×3×10	52.3	81.9	65.1	89.9
GTA	2D R50	8×1×1	50.6	78.8	63.5	88.6
GTA	2D R50	16×1×1	52.0	80.5	64.7	89.3
GTA	2D R50+TSM	8×1×1	51.6	79.8	63.7	88.9
GTA	2D R50+TSM	16×1×1	53.7	81.7	65.3	89.6
$\mathbf{GTA}_{En}$	2D R50+TSM	(16+8)×3×2	**56.5**	<u>83.1</u>	**68.1**	**91.1**

Table 3.8: Comparisons with state-of-the-art methods on Something-Something v1 & v2 datasets. **Bold** and <u>underline</u> shows the highest and second highest results.

tion illustrates the comparison of our GTA and NL added to 2D and 3D CNN backbones. We can see that GTA achieves consistent improvement over the NL counterpart on 2D and 3D CNN backbones. And adding GTA to SlowFast-R101 can achieve 79.8% top-1 accuracy on Kinetics-400 dataset, which is the state-of-the-art performance on Kinetics-400.

Something-Something v1&v2 We compare our approach with the state-of-the-art methods on SSv1 & SSv2 datasets. As shown in Table 3.8, given 8 input frames, our approach based on 2D RestNet-50 with TSM backbone achieves 51.6% and 63.7% on SSv1 and SSv2 at top-1 accuracy, respectively. Specifically, with the same number of input frames, our approach outperforms TRN [75] which utilizes relation networks, and

Method	Backbone	Frames	SSv1	SSv2
$\text{TRN}_{\text{RGB+Flow}}$ [75]	BNInc	8+8	40.7	56.2
TSM [11]	2D R50	16	46.0	64.3
$\text{TSM}_{\text{RGB+Flow}}$ [11]	2D R50	16+16	**50.7**	<u>66.6</u>
STM [40]	2D R50	16	43.1	63.5
bLVNet-TAM [113]	2D R101	64	48.9	-
ECO_{En}*Lite* [76]	BNInc+3D R18	92	42.3	-
I3D+NL+GCN [114]	3D R50	32	45.0	-
TEA [96]	3D R50	16	46.6	63.2
GTA_{En}	2D R50+TSM	16+8	<u>49.8</u>	**66.9**

Table 3.9: Results on the test set of Something-Something v1 & v2 datasets. **Bold** and <u>underline</u> shows the highest and second highest results.

MSNet [44] which incorporates the motion features. This demonstrates that our proposed GTA is more effective in modeling temporal relationships. Our approach also achieves superior results when compared with the recent work that leverages additional modules to improve 3D CNN backbones, such as the non-local block (I3D+NL [10]), GCN (I3D+NL+GCN [114]), the correlation operation (CorrNet [19]), and the multiple temporal aggregation module and the motion excitation module (TEA [96]). Finally, we evaluate the ensemble model (GTA_{En}) by averaging output prediction scores of the 8-frame and 16-frame models, and obtain 56.5% and 68.1% at top-1 accuracy on SSv1 and SSv2, respectively, which achieves the state-of-the-art performance on both datasets.

We also compare the performance of our approach on the test set with the state-of-the-art methods on Something-Something v1 & v2 datasets. As is shown in Table 3.9, our approach based on 2D RestNet-50 with TSM backbone achieves 49.8% and 66.9% on SSv1 and SSv2 at top-1 accuracy, respectively. Although on SSv1 dataset, it is still below the $\text{TSM}_{\text{RGB+Flow}}$, $\text{TSM}_{\text{RGB+Flow}}$ is based on the two-stream network and utilizes additional

optical flow information. With only RGB input, our GTA achieves the best performance among the recently proposed STM [40] and bLVNet-TAM [113] on 2D CNN backbone; I3D+NL+GCN [114] and TEA [96] on 3D CNN backbone.

3.5 Conclusion

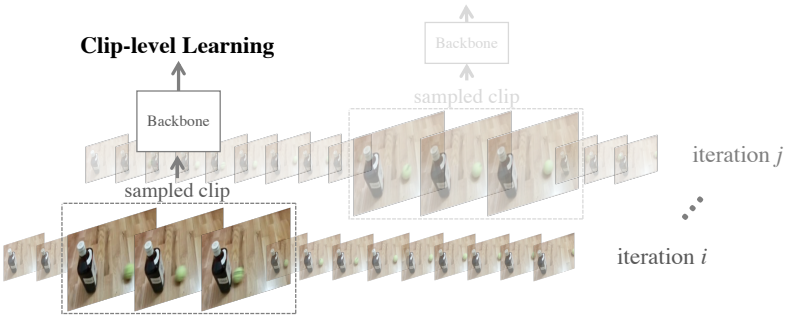
In this chapter, we present Global Temporal Attention (GTA), which is designed for improved temporal modeling in video tasks. GTA is built upon a decoupled self-attention framework, where temporal attention is disentangled from the spatial attention to prevent being dominated by the spatial one. We apply GTA to model the temporal relationships at both pixel-level and region-level. Moreover, GTA directly learns a global, instance-independent attention matrix that generalizes well across different samples. A cross-channel multi-head mechanism is also designed to further improve the temporal modeling in GTA. Experimental results demonstrate that our proposed GTA effectively enhances temporal modeling and achieves state-of-the-art results on three challenging video action benchmarks.

Chapter 4: Beyond Short Clips: End-to-End Video-Level Learning with Collaborative Memories

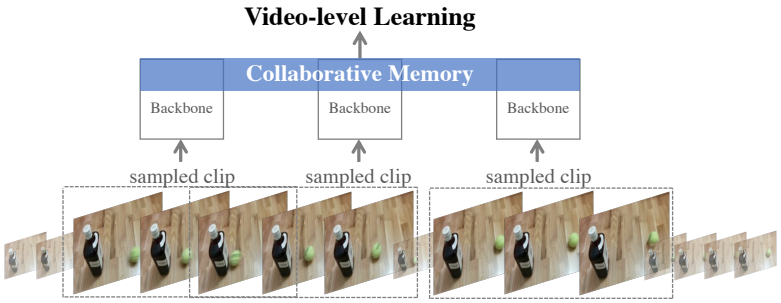
4.1 Introduction

In recent years, end-to-end learning of 3D convolutional networks (3D CNNs) has emerged as the prominent paradigm for video classification [6, 9, 10, 12, 19, 33, 38, 71, 93, 115–119]. Steady improvements in accuracy have come with the introduction of increasingly deeper and larger networks. However, due to their high computational cost and large memory requirements, most video models are optimized at each iteration over short, fixed-length clips rather than the entire video.

Although widely used in modern video models, the clip-level learning framework is sub-optimal for video-level classification. First, capturing long-range temporal structure beyond short clips is not possible as the models are only exposed to individual clips during training. Second, the video-level label may not be well represented in a brief clip, which may be an uninformative segment of the video or include only a portion of the action, as shown in Figure 4.1(a). Thus, optimizing a model over individual clips using video-level labels is akin to training with noisy labels. Recent attempts to overcome these limitations include methods that build a separate network on top of the clip-based back-



(a) Standard: clip-level learning on individual clips at each iteration.



(b) Our framework: video-level learning with collaborative memory.

Figure 4.1: Clip-level learning *vs*. our proposed end-to-end video-level learning framework. (Action label: *something being deflected from something*.)

bone [95, 120, 121]. However, these approaches either cannot be trained end-to-end with the backbone (*i.e*., the video model is optimized over pre-extracted clip-level features) or require ad-hoc backbones which hinder their application in the current landscape of evolving architectures.

In this chapter, we propose an end-to-end learning framework that optimizes the classification model using video-level information collected from multiple temporal locations of the video, shown in Figure 4.1(b). Our approach hinges on a collaborative memory mechanism that accumulates video-level contextual information from multiple clips sampled from the video. Within the same training iteration, this contextual information is shared back with all the clips to enhance the individual clip representations. The collaborative memory allows the model to capture long-range temporal dependencies be-

yond individual short clips by generating clip-specific memories that encode the relation between each local clip and the global video-level context.

Our experiments demonstrate that the proposed training framework is effective and generic. Specifically, our approach does not make any assumption about the backbone architecture. We empirically show that it consistently yields significant gains in accuracy when applied to different state-of-the-art architectures (e.g. SlowFast [12], R(2+1)D [71], I3D-NL [10]). We also introduce and compare several design variants of the collaborative memory. Furthermore, we demonstrate that the accuracy improvements come at a negligible computational overhead and without an increase in memory requirements. Finally, we show that our framework can be extended to action detection where it yields significant improvements without requiring extra information, such as optical flow and object detection predictions, which are commonly used in previous work [122, 123]. We summarize our major contributions as follows:

- A new framework that enables end-to-end learning of video-level dependencies for clip-based models.
- A new collaborative memory mechanism that facilitates information exchange across multiple clips. We explore different design choices and provide insights about the optimization difficulties.
- Experiments demonstrating that our collaborative memory framework generalizes to different backbones and tasks, producing state of the art results for action recognition and detection.

4.2 Related Work

Clip-Level Video Architectures. Since the introduction of 3D CNNs [6, 93, 115] to video classification, new architectures [9, 10, 19, 33, 38, 71, 116–118] have been proposed to learn better spatiotemporal representations. Besides models aimed at improving accuracy, several architectures have been proposed to achieve better performance/cost trade-offs [11, 76, 118, 119, 124–126]. Another line of research involves the design of multiple-stream networks [4, 5, 7, 9, 12, 32, 38], where each stream consumes a different type of input, *e.g.*, RGB or optical flow. Besides CNNs, transformer-based models, *e.g.*, TimeSformer [127], also show promising results. Unlike prior work focused on the design of clip-level architecture, our work proposes a new framework to learn long-range dependencies using existing clip-level models. As we do not make any assumption about the clip-level architecture, our framework generalizes to different backbones and enables end-to-end training of clip models with video-level contextual information.

Video-Level Classification. Several attempts have been made to overcome the limitations of the single-clip training framework. Timeception [95] uses multi-scale temporal convolutions to cover different temporal extents for long-range temporal modeling. Timeception layers are trained on top of a frozen backbone. Both TSN [7] and ECO [76] divide the input video into segments of equal size and randomly sample a short snippet or a single frame from each segment to provide better temporal coverage during training. As the GPU memory cost grows linearly w.r.t. the number of segments, TSN and ECO adopt lightweight 3D CNNs or even 2D CNNs as backbones in order to process multiple seg-

ments simultaneously. These simple backbones limit the performance of the framework. In addition, TSN uses averaging to aggregate the predictions from different segments, whereas we propose a dedicated memory mechanism to model the video-level context. FASTER [121] and SCSampler [128] explore strategies to limit the detrimental impact of applying video-level labels to clips and to save computational cost.

Another related work to our approach is LFB [120]. It leverages context features from other clips to augment the prediction on the current clip. Unlike our approach, context features stored in LFB are *pre-computed* using a *separate* model. As a result, the context features cannot be updated during the training and the model used to extract these context features is not optimized for the task. In contrast, our framework is end-to-end trainable and the accumulated contextual information can back-propagate into the backbone architecture. Note that storing context features is infeasible for large-scale video datasets, *e.g.*, Kinetics, and LFB is mainly designed for action detection applications.

Learning With Memories. Memory mechanisms [129–132] have been widely used in Recurrent Neural Networks for language modeling in order to learn long-term dependencies from sequential text data. Specifically, memory networks [129] have been proposed for question answering (QA), while Sukhbaatar *et al*. [130] have introduced a strategy enabling end-to-end learning of these models. RWMN [133] has extended the QA application on movie videos. Grave *et al*. [131] have proposed to store past hidden activations as a memory that can be accessed through a dot product with the current hidden activation.

These works are similar in spirit to our approach, but our application is in a different domain with different constraints and challenges. Moreover, our collaborative mem-

ory mechanism is designed to capture the interactions among the samples, is extremely lightweight and memory-friendly, and is suitable for training computationally intensive video models.

4.3 End-to-End Video-Level Learning with Collaborative Memory

We start with an overview of the proposed framework, then present different designs for the collaborative memory. We conclude with a discussion of implementation strategies to cope with the GPU memory constraint.

4.3.1 Overview of the Proposed Framework

Given a video recognition architecture (*e.g.*, I3D [9], R(2+1)D [71], SlowFast [12]) that operates on short, fixed-length clips, our goal is to perform video-level learning in an end-to-end manner. In particular, we aim to optimize the *clip-based* model using *video-level* information collected from the whole video. To achieve this, we start by sampling multiple clips from the video within the same training iteration in order to cover different temporal locations of the video. The clip-based representations generated from multiple clips are then accumulated via a collaborative memory mechanism that captures interactions among the clips and builds video-level contextual information. After that, clip-specific memories are generated to enhance the individual clip-based representations by infusing the video-level information into the backbone. Finally, the sampled clips are jointly optimized with a video-level supervision applied to the consensus of predictions from multiple clips.

Multi-clip sampling. Given a video $\mathcal{V} = \{I_0, ..., I_{T-1}\}$ with T frames, we sample N clips $\{\mathcal{C}_0, ..., \mathcal{C}_{N-1}\}$ from the video at each training iteration. Each short clip $\mathcal{C}_n = \{I_{t_n}, ..., I_{t_n+L-1}\}$ consists of L consecutive frames randomly sampled from the full-length video where t_n indicates the index of the start frame. N is a hyper-parameter that can be decided based on the ratio between the video length and the clip length to ensure sufficient temporal coverage. The sampled clips are then fed to the backbone to generate clip-based representations $\{X_n\}_{n=0}^{N-1}$, where $X_n = f(\mathcal{C}_n)$, and f represents the clip-level backbone. In the traditional clip-level classification, X_n is directly used to perform the final prediction via a classifier h : $\mathbf{y}_n = h(X_n) = h(f(\mathcal{C}_n))$, where $\mathbf{y}_n$ is the vector of classification scores.

Collaborative memory. Our approach hinges on a collaborative memory mechanism that accumulates information from multiple clips for learning video-level dependencies and then shares this video-level context back with the individual clips, as illustrated in Figure 4.2. Specifically, the collaborative memory involves two stages:

- *Memory interactions:* A global memory of the whole video is constructed by accumulating information from multiple clips: $\mathcal{M} = Push(\{X_n\}_{n=0}^{N-1})$. The global memory is then shared back with the individual clips in order to generate clip-specific memories: $M_n = Pop(\mathcal{M}, X_n)$.
- *Context infusion:* The individual clip-based representations are infused with video-level context. This is done by means of a gating function g that enhances each clip representation with the information stored in the clip-specific memory: $\hat{X}_n = g(X_n, M_n)$.

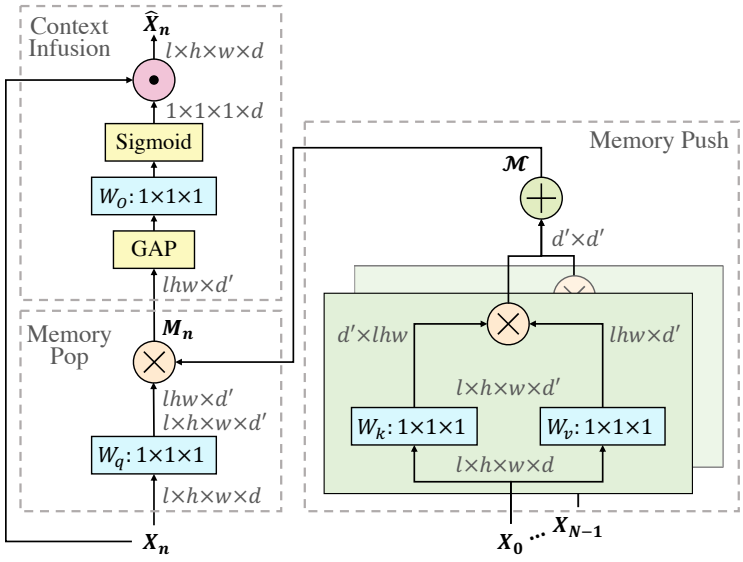


Figure 4.2: Collaborative memory with associative memory and feature gating. The feature maps are shown as the shape of their tensors, *e.g.*, $l \times h \times w \times d$. GAP denotes global average pooling. $\otimes$ and $\odot$ indicate matrix and elementwise multiplication, respectively.

Video-level supervision. To facilitate the joint optimization over multiple clips, we apply a video-level loss that takes into account the clip-level predictions as well as the video-level prediction aggregated from all N sampled clips. Formally, we first aggregate the clip-level predictions via average pooling over N clips: $H = \frac{1}{N}\sum_{n=0}^{N-1} h(\hat{X}_n) = \frac{1}{N}\sum_{n=0}^{N-1} h(g(f(\mathcal{C}_n), M_n))$. Then our video-level loss can be written as

$$\mathcal{L}_{video} = \frac{1}{N}\sum_{n=0}^{N-1}\mathcal{L}(h(\hat{X}_n)) + \alpha\mathcal{L}(H). \tag{4.1}$$

$\mathcal{L}$ denotes the cross-entropy loss for classification and α is the weight to balance the two terms which account for the clip-level losses and the video-level loss aggregated from all N clips. All the parameters (*i.e.*, f, g, and h) are optimized end-to-end w.r.t. this objective.

4.3.2 Collaborative memory

Our idea of collaborative memory is generic and can be implemented in a variety of ways. In this section we introduce a few possible designs. We empirically evaluate these different options in Section 4.4.3.

Memory interactions. The design of memory interactions should follow two principles: 1) The memory footprint for storing the global memory should be manageable; 2) Interactions with the memory should be computationally efficient. The first principle implies that the memory consumption should not grow with the number of clips N. Thus, simply storing all clip-based features is not feasible.

– Average Pooling: Let the clip-based representation X_n be a $k \times d$ matrix, where k is the spatial-temporal resolution (*i.e.*, $height \times width \times length$) and d is the number of channels. A simple strategy is to perform a global average pooling over all sampled clips: $\mathcal{M} = Push(\{X_n\}_{n=0}^{N-1}) = Pool(\{X_n W_I\}_{n=0}^{N-1})$. $W_I \in \mathbb{R}^{d \times d'}$ is a learnable weight matrix to reduce the dimensionality from d to d'. This global information can be simply shared back with all the clips: $M_n = Pop(\mathcal{M}, X_n) = \mathcal{M}$.

– Associative Memory: Although avg/max pooling is capable of collecting information from multiple clips, it fails to capture the inter-clip dependencies and the clip-specific information cannot be retrieved from the global memory $\mathcal{M}$. This motivates us to design a new mechanism that enables the retrieval of clip-specific information when needed. Inspired by associative networks [132, 134], we propose to accumulate the clip-

level features using the outer product operator to generate the global memory $\mathcal{M}$:

$$Push(\{X_n\}_{n=0}^{N-1}) = \frac{1}{N}\sum_{n=0}^{N-1}(X_nW_k)^T(X_nW_v). \tag{4.2}$$

Given the n-th clip, we obtain its clip-specific memory by:

$$M_n = Pop(\mathcal{M}, X_n) = (X_nW_q)\mathcal{M}, \tag{4.3}$$

where $W_k, W_v, W_q \in \mathbb{R}^{d\times d'}$ are learnable weight matrices for memory interactions and dimension reduction. Note that this memory design can be viewed as implementing a form of video-level inter-clip attention, where the clip-based representation X_n attends to features generated from all sampled clips of the video in proportion to their similarities: $M_n = \frac{1}{N}\sum_{m=0}^{N-1}\left[(X_nW_q)(X_mW_k)^T\right](X_mW_v)$. However, unlike the self-attention mechanism [10, 83], our design is more efficient in both computation and memory consumption as it does not require to store all clip-level features or perform pairwise comparison between all the clips.

Context Infusion. One way to incorporate the clip-specific memory M_n with the clip-level features X_n is through a residual connection: $\hat{X}_n = M_nW_O + X_n$, where $W_O \in \mathbb{R}^{d'\times d}$ is a linear transformation to match the feature dimensionality. However, as we will show in our experiments (Figure 4.5), this design tends to overfit to the clip-specific memory during training and leads to inferior performance. As M_n stores much more information than a single clip-level feature X_n, the model mostly relies on M_n during training and makes little use of X_n.

In light of the above observation, we propose to infuse context information into the clip-level features through a *feature gating* operation. Rather than allowing the model to directly access the clip-specific memory M_n, feature gating forces the model to recalibrate the strengths of different clip-level features using the contextual information. Formally, the enhanced features are computed as

$$\hat{X}_n = \Big(J + \sigma(\hat{M}_n W_O)\Big) \odot X_n, \tag{4.4}$$

where σ is the sigmoid activation function, $\odot$ is the elementwise multiplication and J is an all-ones matrix for residual connection. $\hat{M}_n$ is obtained by averaging the spatial and temporal dimensions of M_n: $\hat{M}_n = GAP(M_n)$. Our feature gating operation can be considered as a channel-wise attention mechanism similar to context gating [38, 135] and the SE block [85]. However, the attention weights in our method are generated by video-level contextual information, instead of self-gating values that capture channel-wise relationships within the same clip. Experimental results show that our proposed feature gating design alleviates the optimization difficulties during training and enables a more effective use of the video-level contextual information.

4.3.3 Coping with the GPU Memory Constraint

A challenge posed by video-level learning is the need to jointly optimize over multiple clips under a fixed and tight GPU memory budget. In this section, we discuss two simple implementations that allow end-to-end training of video-level dependencies under this constraint.

Batch reduction. Let B be the size of the mini-batch of videos used for traditional clip-level training. Our approach can be implemented under the same GPU memory budget by reducing the batch size by a factor of N: $\hat{B} = round(B/N)$. This allows us to load into the memory N clips for each of the B/N different videos. In order to improve the clip diversity for updating the batch-norm [63] parameters within a mini-batch, we propose to calculate batch-norm statistics using only the clips from different videos. Although this implementation cannot handle arbitrarily large N, it is simple, efficient and we found it applicable to most settings in practice. For example, a typical choice of batch size for training clip-based models is a 8-GPU machine with $B = 64$; our approach can be implemented under this memory setup by using in each iteration $\hat{B} = 16$ different videos, and by sampling from each of them $N = 4$ clips.

Multi-iteration. Instead of directly loading N clips into one mini-batch, we can also unroll the training of a video into N iterations. Each iteration uses one of the N clips. This implementation is memory-friendly and consumes the same amount of memory as the standard single-clip training framework. It allows us to process arbitrarily long videos with arbitrarily large N. When incorporating the collaborative memory, we simply perform a two-scan process: the first scan generates the clip-based features to build the global memory $\mathcal{M}$ and the second scan generates the classification output of each clip conditioned on $\mathcal{M}$.

4.4 Experiments

To demonstrate the advantages of our end-to-end video-level learning framework, we conduct extensive experiments on four action recognition benchmarks with different backbone architectures. We implement our models and conduct the experiments using the PySlowFast codebase [136].

4.4.1 Experimental Setup

Datasets. Kinetics [88] (K400 & K700) is one of the most popular datasets for large-scale video classification. Charades [137] is a multi-label dataset with long-range activities. Something-Something-V1 [68] is a dataset requiring good use of temporal relationships for accurate recognition. Following the standard protocol, we use the training set for training and report top-1 accuracy on the validation set.

Backbones. We evaluate our framework using different backbone architectures including I3D [10], R(2+1)D [19,71], Slow-only [12] and SlowFast [12], optionally augmented with non-local blocks (NL) [10]. We attach the proposed collaborative memory to the last convolutional layer of these backbones for joint training.

Training. We first train the backbones by themselves following their original schedules [12, 19], then re-train the backbones in conjunction with our collaborative memory for video-level learning. When training on Kinetics, we use synchronous SGD with a cosine learning schedule [74]. For Charades and Something-Something-V1, we follow the recipe from PySlowFast [136] and initialize the network weights from the models

pre-trained on Kinetics, since these two datasets are relatively small. For the video-level training, we employ the batch reduction strategy to handle the GPU memory constraint by default and apply the linear scaling rule [73] to adjust the training schedule accordingly.

Inference. Following [10, 12], we uniformly samples 3×10 crops from each video for testing (*i.e.*, 3 spatial crops and 10 temporal crops). The global memory $\mathcal{M}$ is aggregated from 10 spatially centered crops and shared for the inference of all 30 crops. We employ the multi-iteration approach from Section 4.3.3 during inference to overcome the GPU memory constraint. The softmax scores of all 30 clips are averaged for the final video-level prediction.

4.4.2 Evaluating Collaborative Memory

For all the experiments in this section we use the associative version of the collaborative memory with feature gating since, as demonstrated in ablation studies (Section 4.4.3), this design provides the best results.

Effectiveness of video-level learning. We begin by presenting an experimental comparison between our proposed video-level learning and the standard clip-level training applied to the same architecture. For this evaluation we use the Slow-only backbone with 50 layers, which can be considered as a 3D ResNet [34]. In order to investigate the impact of temporal coverage on video-level learning, we train models using different numbers of sampled clips per video: $N \in \{1, 3, 5, 7, 9\}$. $N = 1$ corresponds to the conventional clip-level training, as we only sample one clip per video. In such case the collaborative

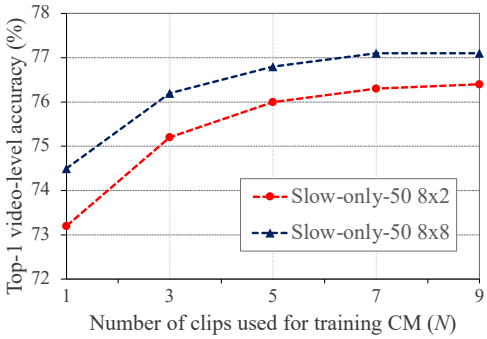


Figure 4.3: **Video-level accuracy** on Kinetics-400 with CM. The horizontal axis shows the number of clips (N) used for training CM. Note that all models use 30 crops for inference.

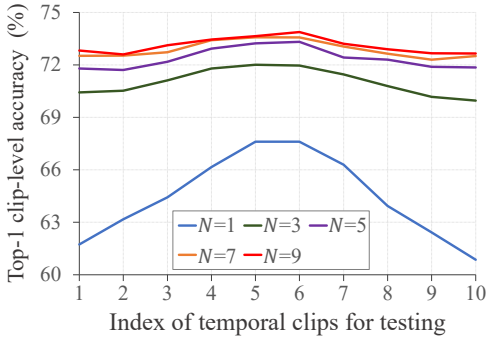


Figure 4.4: **Clip-level accuracy** on Kinetics-400 at 10 different temporal locations within the video. N indicates the number of clips used for training CM.

memory (CM) is limited to perform "self-attention" within the single clip. For $N > 1$, CM captures video-level information across the N clips.

Figure 4.3 shows the *video*-level accuracy achieved by the different models, all using the same testing setup of 3×10 crops per video. Note that under this setting all models "look" at the same number of clips for each video in testing. As shown in Figure 4.3, our CM framework significantly improves the video-level accuracy. For example, when the clip length is 8×8 (8 frames with a temporal stride of 8), using CM with $N = 9$ yields a remarkable 2.6% improvement compared with training using a single clip (74.5% *vs.* 77.1%). When the clip has a shorter length (*i.e.*, 8×2), our method gives an even larger gain, 3.2% (73.2% *vs.* 76.4%). As expected, the improvement from our method increases with the number of sampled clips N. The performance saturates when $N \geq 7$. To keep the training time more manageable, we use $N = 5$ by default.

Figure 4.4 shows the clip-level accuracy at different temporal locations of a video, according to their temporal order. When $N = 1$, the clips from the middle of the video have much higher accuracy than the clips from the beginning or the end of the video, as

Model	Baseline	Ours	△	FLOPs
Slow-only-50 8×8 [12]	74.4	**76.8**	+2.4	1.03×
I3D-50+NL 32×2 [10]	74.9	**77.5**	+2.4	1.02×
R(2+1)D-50 16×2 [71]	75.7	**78.0**	+2.3	1.01×
SlowFast-50 4×16 [12]	75.6	**77.8**	+2.2	1.02×
SlowFast-50 8×8 [12]	76.8	**78.9**	+2.1	1.03×

Table 4.1: Generalization to **different backbone architectures**. We report the video-level accuracy on Kinetics-400 for both standard clip-level training ("Baseline") and video-level training with CM ("Ours").

the middle clips tend to include more relevant information. CM significantly improves the clip-level accuracy by augmenting each clip with video-level context information (*i.e.*, $N \geq 3$), especially for clips near the boundary of the video. This is a clear indication that our memory mechanism is capable of capturing video-level dependencies and sharing them effectively with the clips within the video to boost the recognition accuracy.

Generalization to different backbones. As we do not make any assumption about the backbone, our video-level end-to-end learning framework can be seamlessly integrated with different architectures and input configurations (*e.g.*, clip length, sampling stride, *etc*.). As shown in Table 4.1, CM produces consistent video-level accuracy gains of over 2% on top of state-of-the-art video models, including I3D with non-local blocks [10], the improved R(2+1)D network [19, 71] and the recent SlowFast network [12]. Note that we achieve these improvements with only negligible additional inference cost, about 1-3% more FLOPs compared to the backbone themselves.

Multi-clip	Memory	End-to-end	Top-1
	✓	✓	74.5
✓		✓	75.5
✓	✓		75.9
✓	✓	✓	**76.8**

Table 4.2: Evaluating **different components** of our video-level learning framework.

4.4.3 Ablation Studies

We conduct ablation experiments on Kinetics-400. Top-1 video-level accuracy (%) is reported. Unless otherwise stated, we use Slow-only [12] with 50 layers and the input clip length is 8×8. R(2+1)D is also 50 layers with a clip length of 16×2.

Assessing the components in our framework. Unlike most prior work on video-level modeling [95, 120], our framework is end-to-end trainable. To show the benefits of end-to-end learning in improving the backbone, we conduct an ablation that freezes the parameters of the backbone and only updates the parameters from the collaborative memory and the FC layers for classification. As shown in Table 4.2, end-to-end learning improves the performance by 1.1% compared with learning video-level aggregation on top of the frozen backbones (76.8% *vs*. 75.9%).

Table 4.2 also shows the result of video-level learning without using CM. This is done by optimizing multiple clips per video but without sharing any information across the clips. Interestingly, this simple setup also delivers a good improvement over the single-clip learning baseline (75.5% vs. 74.5%). The gain comes from the joint optimization over multiple clips of a video, which again confirms the importance of video-level learning for classification. Our CM framework achieves the best performance with all the

Setting	Associative	Gating	Top-1
Multi-clip (w/o memory)			75.5
CM (avgpool)		✓	75.8
CM (residual)	✓		76.0
CM (default)	✓	✓	**76.8**

Table 4.3: Comparing **different designs** of our collaborative memory mechanism.

Setting	#Param.	Top-1
$\alpha = 1$	49.2 M	76.8
$\alpha = 2$	40.9 M	76.8
$\alpha = 4$	36.7 M	**76.8**
$\alpha = 8$	34.6 M	76.4

Table 4.4: Varying **channel reduction ratio** $\alpha = d/d'$.

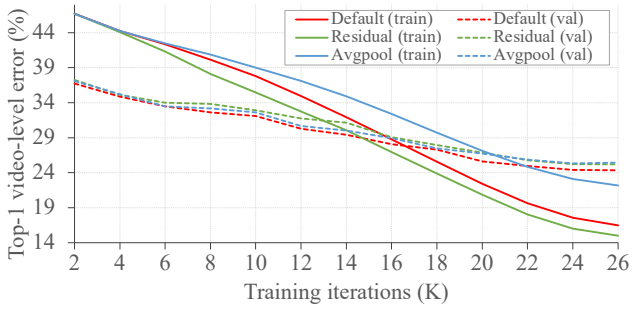


Figure 4.5: Video-level training/validation errors on Kinetics-400 for different designs of the collaborative memory.

components enabled.

Collaborative memory design. Our default design uses the associative memory for memory interactions and a feature gating operation for context infusion. In Table 4.3, we explore other design choices by replacing the associative memory with average pooling or substituting the feature gating with a residual connection.

We observe that all these variants offer improvement over the naïve video-level learning setup without the memory. However, the gain provided by average pooling is relatively small, which is not surprising given that there is no the inter-clip interaction. While we also witness a performance drop by removing the feature gating operation, the reason behind it is different. As shown by the training/validation error curves in Figure 4.5, the model without feature gating achieves lower training error but higher valida-

Model	Stage-wise	Top-1
Slow-only		76.1
	✓	**76.8**
R(2+1)D		77.7
	✓	**78.0**

Table 4.5: **Stage-wise training** *vs*. training everything from scratch.

Model	Batch reduction	Multi-iteration	Top-1
Slow-only		✓	76.6
	✓		**76.8**
R(2+1)D		✓	77.9
	✓		**78.0**

Table 4.6: Comparing different ways of training CM: **batch reduction** *vs*. multi-iteration.

tion error. This suggests that the model degenerates during training due to over-fitting to the video-level context.

In Table 4.4 we ablate the number of channels used in CM (d' in Eq. 4.2, 4.3), which can be controlled by the channel reduction ratio $\alpha = d/d'$. We can see that the results remain unchanged as long as the reduction ratio is reasonable ($\alpha \leq 4$). We use $\alpha = 4$ as the default value in our experiments since it introduces fewer extra parameters.

Training strategies. Recall that we adopt a stage-wise training strategy: the backbones are first trained using standard clip-level training recipes and then re-trained in conjunction with CM for video-level learning. In Table 4.5, we compare this strategy with training everything from scratch (equivalent training iterations are used for both strategies). Experiments on two different backbones show that training everything from scratch yields slightly worse results. We hypothesize that stage-wise training allows the optimization in the second stage to focus on effective long-range modeling thanks to the well-initialized backbone. We note that non-local networks are also trained in a stage-wise way.

We also compare the two methods to cope with the GPU memory constraint (Section 4.3.3). As shown in Table 4.6, the accuracy of the two methods is almost the same and the difference is within the margin of randomness, which makes sense as the two

Model	Temporal stride				CM
	2	**4**	**8**	**16**	
Slow-only	73.2	74.3	74.4	74.4	**76.8**
R(2+1)D	75.7	76.4	75.0	72.2	**78.0**

Table 4.7: Comparing CM with backbones using clips with **large temporal strides**.

methods are technically identical.

Limitations of temporal striding. One simple way to increase the temporal coverage of a video model is to use larger temporal strides when sampling the frames of the input clips. We compare our video-level learning framework with this strategy in Table 4.7. Note that we keep the temporal strides of CM the same as the original backbones, *i.e.*, 8 frames for Slow-only and 2 frames for R(2+1)D.

We can see that increasing the temporal coverage through striding yields only a modest gain in accuracy. Notably, using a very large stride even hurts performance for some architectures like R(2+1)D. In contrast, our approach can learn long-range dependencies and improves the performances of short clip-based backbones by large margins.

4.4.4 Comparison with the State of the Art

Previous experimental results are from Kinetics-400. To demonstrate that our method can generalize to different datasets, we further evaluate our method on Kinetics-700 [140], Charades [137] and Something-Something-V1 [68]. Among them, Charades has longer-range activities (30 seconds on average), whereas Something-Something-V1 includes mostly human-object interactions. We compare the results with the state of the art in Table 4.8, 4.9, 4.10 and 4.11. Our proposed CM framework yields consistent gains of

Methods	Pretrain	Only RGB	GFLOPs × crops	Top-1
I3D [9]	ImageNet	✗	216×N/A	75.7
S3D-G [38]	ImageNet	✗	142.8×N/A	77.2
LGD-3D-101 [70]	ImageNet	✗	N/A	81.2
I3D-101+NL [10]	ImageNet	✓	359×30	77.7
ip-CSN-152 [119]	Sports1M	✓	109×30	79.2
CorrNet-101	Sports1M	✓	224×30	81.0
MARS+RGB [78]	none	✓	N/A	74.8
DynamoNet [33]	none	✓	N/A	77.9
CorrNet-101 [19]	none	✓	224×30	79.2
SlowFast-101 8×8 [12]	none	✓	106×30	77.9
SlowFast-101 16×8 [12]	none	✓	213×30	78.9
SlowFast-101+NL 16×8 [12]	none	✓	234×30	79.8
Ours (R(2+1)D-101 32×2)	none	✓	243×30	80.5
Ours (SlowFast-101 8×8)	none	✓	128×30	80.0
Ours (SlowFast-101+NL 8×8)	none	✓	137×30	**81.4**

Table 4.8: Comparison with the state-of-the-art on Kinetics-400.

Methods	Pretrain	GFLOPs × crops	Top-1
SlowFast-101+NL 8×8 [12]	K600	115×30	70.6
SlowFast-101+NL 16×8 [12]	K600	234×30	71.0
SlowFast-50 4×16*	K600	36×30	66.1
SlowFast-101 8×8*	K600	126×30	69.2
SlowFast-101+NL 8×8*	K600	135×30	70.2
Ours (SlowFast-50 4×16)	K600	37×30	68.3
Ours (SlowFast-101 8×8)	K600	128×30	70.9
Ours (SlowFast-101+NL 8×8)	K600	137×30	**72.4**

Table 4.9: Comparison with the state-of-the-art on Kinetics-700. * indicates results reproduced by us.

over 2% for different variants of SlowFast on all datasets. These improvements are very significant given that SlowFast is among the best video backbones.

On Kinetics-400 and Kinetics-700, our method establishes a new state of the art, as shown in Table 4.8 and 4.9. Notably, we achieve these results without pre-training on other datasets or using optical flow. Similarly, our method outperforms the state of

Methods	Pretrain	GFLOPs × crops	Top-1
TRN [75]	ImageNet	N/A	25.2
I3D-101+NL [10]	ImageNet+K400	544 × 30	37.5
STRG [114]	ImageNet+K400	630 × 30	39.7
Timeception [95]	K400	N/A	41.1
LFB (I3D-101+NL) [120]	K400	N/A	42.5
SlowFast-101+NL [12]	K400	234×30	42.5
AVSlowFast-101+NL [138]	K400	278×30	43.7
SlowFast-50 16×8*	K400	131×30	39.4
SlowFast-101+NL 16×8*	K400	273×30	41.3
Ours (SlowFast-50 16×8)	K400	135×30	42.9
Ours (SlowFast-101+NL 16×8)	K400	277×30	**44.6**

Table 4.10: Comparison with the state-of-the-art on Charades. * indicates results reproduced by us.

Methods	Pretrain	Only RGB	Top-1
S3D-G [38]	ImageNet	✗	48.2
ECO [76]	none	✗	49.5
Two-stream TSM [11]	ImageNet	✗	52.6
MARS+RGB+Flow [78]	K400	✗	53.0
NL I3D-50+GCN [114]	ImageNet	✓	46.1
GST-50 [139]	ImageNet	✓	48.6
MSNet [44]	ImageNet	✓	52.1
CorrNet-101 [19]	Sports1M	✓	53.3
SlowFast-50 8×8*	K400	✓	50.1
SlowFast-101+NL 8×8*	K400	✓	51.2
Ours (SlowFast-50 8×8)	K400	✓	52.3
Ours (SlowFast-101+NL 8×8)	K400	✓	**53.7**

Table 4.11: Comparison with the state-of-the-art on Something-Something-V1. * indicates results reproduced by us.

the art on both Charades (in Table 4.10) and Something-Something-V1 (in Table 4.11). On Charades, our CM framework yields more than 3% gains (*e.g.*, 44.6% vs 41.3%). This demonstrates that CM performs even better on datasets that have longer videos and require longer-term temporal learning. Note that our method significantly outperforms other recent work on long-range temporal modeling (*e.g.*, Timeception [95], LFB [120]).

4.4.5 Collaborative Memory for Action Detection

In this section, we show that the benefits of our framework also extend to the task of action detection. We evaluate our method on AVA [141], which includes 211k training and 57k validation video segments. AVA v2.2 provides more consistent annotations than v2.1 on the same data. We report mean average precision (mAP) over 60 frequent classes on the validation set following the standard protocol.

Adaptation to action detection. Adapting our approach to action detection is straightforward. Instead of randomly sampling multiple clips from the whole video, we sample clips within a certain temporal window $t_n \in [t - w, t + w]$ to detect action at time t, where t_n indicates the center frame of the nth sampled clip and $2w + 1$ is the window size. As AVA includes sparse annotations at one frame per second, we simply sample the clips with a one-second stride such that the sampled clips are centered at frames with annotations. In this way, the temporal window size increases accordingly as we use a larger number of clips N during training. After that, we jointly optimize these sampled clips with their own annotations. The collaborative memory is used to share long-range context information among sampled clips. We use $N = 9$ in our experiments and follow the schedule in AIA [123] for model training.

Quantitative results. We compare CM with the state of the art on AVA in Table 4.12. Although the CM framework is not specifically designed for action detection, it achieves results comparable with the state of the art. In particular, CM yields a consistent improvement of more than 2% for different backbone networks (*e.g.*, 2.9% gain for SlowFast-

Methods	Pretrain	mAP
ACRN [122]	K400	17.4†
AVSF-50 4×16 [138]	K400	27.8†
AT (I3D) [142]	K400	25.0
LFB(R50+NL) [120]	K400	25.8
R50+NL* [120]	K400	23.6
SF-50 4×16* [12]	K400	23.6
Ours (R50+NL)	K400	26.3
Ours (SF-50 4×16)	K400	25.8

Methods	Pretrain	mAP
AVSF-101 8×8 [138]	K400	28.6†
AIA(SF-50 4×16) [123]	K700	29.8†
AIA(SF-101 8×8) [123]	K700	32.3†
SF-101+NL 8×8 [12]	K600	29.0
SF-50 4x16* [12]	K700	26.9
SF-101 8x8* [12]	K700	29.0
Ours (SF-50 4×16)	K700	29.8
Ours (SF-101 8×8)	K700	31.6

Table 4.12: Comparison with SOTA on AVA v2.1 (left) and v2.2 (right). † indicates results with extra information other than RGB frames, such as optical flow, audio and objection detection predictions. * indicates results reproduced by us.

50 4×16 backbone on AVA v2.2). This demonstrate that we can effectively extend our method to the detection task and achieve significant improvements as well. Note that our method also outperforms LFB [120] when using the same backbone (*i.e.*, R50-I3D+NL) on AVA v2.1 (26.3% *vs*. 25.8%).

4.5 Conclusions

We have presented an end-to-end learning framework that optimizes classification models using video-level information. Our approach hinges on a novel collaborative memory mechanism that accumulates contextual information from multiple clips sampled from the video and then shares back this video-level context to enhance the individual clip representations. Long-range temporal dependencies beyond short clips are captured through the interactions between the local clips and the global memory. Extensive experiments on both action recognition and detection benchmarks show that our framework significantly improves the accuracy of video models at a negligible computational overhead.

Chapter 5: Spatio-Temporal Progressive Learning for Video Action Detection

5.1 Introduction

Spatio-temporal action detection aims to recognize the actions of interest that present in a video and localize them in both space and time. Inspired by the advances in the field of object detection in images [143, 144], most recent work approaches this task based on the standard two-stage framework: in the first stage action proposals are produced by a region proposal algorithm or densely sampled anchors, and in the second stage the proposals are used for action classification and localization refinement.

Compared to object detection in images, spatio-temporal action detection in videos is however a more challenging problem. New challenges arise from both of the above two stages when the temporal characteristic of videos is taken into account. First, an action tube (i.e., a sequence of bounding boxes of action) usually involves spatial displacement over time, which introduces extra complexity for proposal generation and refinement. Second, effective temporal modeling becomes imperative for accurate action classification, as a number of actions are only identifiable when temporal context information is available.

Previous work usually exploits temporal information by performing action detection at the clip (i.e., a short video snippet) level. For instance, [14, 145] take as input a sequence of frames and output the action categories and regressed tubelets of each clip. In order to generate action proposals, they extend 2D region proposals to 3D by replicating them over time, assuming that the spatial extent is fixed within a clip. However, this assumption would be violated for the action tubes with large spatial displacement, in particular when the clip is long or involves rapid movement of actors or camera. Thus, using long cuboids directly as action proposals is not optimal, since they introduce extra noise for action classification and make action localization more challenging, if not hopeless. Recently, there are some attempts to use adaptive proposals for action detection [146, 147]. However, these methods require an offline linking process to generate the proposals.

To this end, we present a novel learning framework, **S**patio-**TE**mporal **P**rogressive (STEP) action detector, for video action detection. As illustrated in Figure 5.1, unlike existing methods that directly perform action detection in one run, our framework involves a multi-step optimization process that progressively refines the initial proposals towards the final solution. Specifically, STEP consists of two components: **spatial refinement** and **temporal extension**. Spatial refinement starts with a small number of coarse-scale proposals and updates them iteratively to better classify and localize action regions. We carry out the multiple steps in a sequential order, where the outputs of one step are used as the proposals for next step. This is motivated by the fact that the regression outputs can better follow actors and adapt to action tubes than the input proposals. Temporal extension focuses on improving classification accuracy by incorporating longer-range temporal

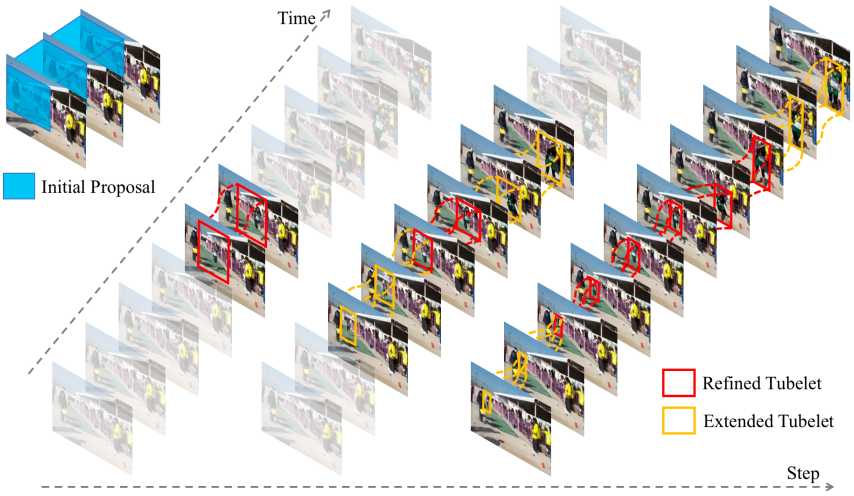


Figure 5.1: A schematic overview of spatio-temporal progressive learning for action detection. Starting with a coarse-scale proposal cuboid, it progressively refines the proposal towards the action, and adaptively extends the proposal to incorporate more related temporal context at each step.

information. However, simply taking a longer clip as input is inefficient and also ineffective since a longer sequence tends to have larger spatial displacement, as shown in Figure 5.1. Instead, we progressively process longer sequences at each step and adaptively extend proposals to follow action movement. In this manner, STEP can naturally handle the spatial displacement problem and therefore provide more efficient and effective spatio-temporal modeling. Moreover, STEP achieves superior performance by using only a handful (e.g., 11) of proposals, obviating the need to generate and process large numbers (e.g., $>$1K) of proposals due to the tremendous spatial and temporal search space.

To our knowledge, this work provides the first end-to-end progressive optimization framework for video action detection. We bring up the spatial displacement problem in action tubes and show that our method can naturally handle the problem in an efficient and effective way. Extensive evaluations find our approach to produce superior detection results while only using a small number of proposals.

5.2 Related Work

Spatio-Temporal Action Detection. Inspired by the recent advances in image object detection, a number of efforts have been made to extend image object detectors (e.g., R-CNN, Fast R-CNN and SSD) to the task as frame-level action detectors [148–154]. The extensions mainly include: first, optical flow is used to capture motion cues, and second, linking algorithms are developed to connect frame-level detection results as action tubes. Although these methods have achieved promising results, the temporal property of videos is not explicitly or fully exploited as the detection is performed on each frame independently. To better leverage the temporal cues, several recent work has been proposed to perform action detection at clip level. For instance, ACT [145] takes as input a short sequence of frames (e.g., 6 frames) and outputs the regressed tubelets, which are then linked by a tubelet linking algorithm to construct action tubes. Gu et al. [14] further demonstrate the importance of temporal information by using longer clips (e.g., 40 frames) and taking advantage of I3D pre-trained on the large-scale video dataset [9]. Rather than linking the frame or clip level detection results, there are also some methods that are developed to link the proposals before classification to generate action tube proposals [146, 147].

Progressive Optimization. This technique has been explored in a range of vision tasks from pose estimation [155], image generation [156] to object detection [157–160]. Specifically, the multi-region detector [158] introduces iterative bounding box regression with R-CNN to produce better regression results. AttractioNet in [159] employs a multi-stage procedure to generate accurate object proposals that are then input to Fast R-CNN. G-

CNN [160] trains a regressor to iteratively move a grid of bounding boxes towards objects. Cascade R-CNN [157] proposes a cascade framework for high-quality object detection, where a sequence of R-CNN detectors are trained with increasing IoU thresholds to iteratively suppress close false positives.

5.3 Method

In this section, we introduce the proposed progressive learning framework STEP for video action detection. We first formulate the problem and provide an overview of our approach. We then describe in details the two primary components of STEP including spatial refinement and temporal extension. Finally, the training algorithm and implementation details are presented.

5.3.1 Framework Overview

Proceeding with the recent work [14, 145], our approach performs action detection at clip level, i.e., detection results are first obtained from each clip and then linked to build action tubes across a whole video. We assume that each action tubelet of a clip has a constant action label, considering the short duration of a clip, e.g., within one second.

Our target is to tackle the action detection problem through a few progressive steps, rather than directly detecting actions all at one run. In order to detect the actions in a clip I_t with K frames, according to the maximum progressive steps S_{max}, we first extract the convolutional features for a set of clips $\mathcal{I} = \{I_{t-S_{max}+1}, ..., I_t, ..., I_{t+S_{max}-1}\}$ using a backbone network such as VGG16 [161] or I3D [9]. The progressive learning starts

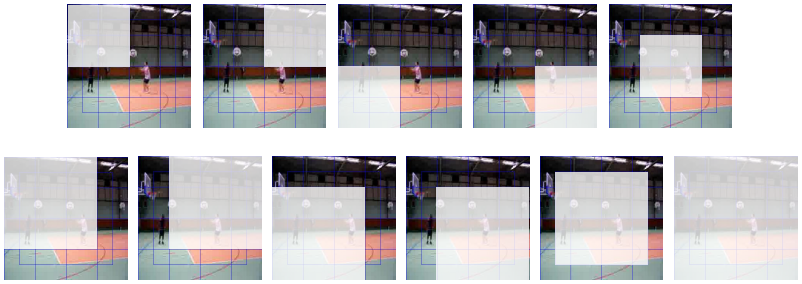


Figure 5.2: Example of the 11 initial proposals: 2D boxes are replicated across time to obtain cuboids.

with M pre-defined proposal cuboids $B^0 = \{b_i^0\}_{i=1}^M$ and $b_i^0 \in \mathcal{R}^{K\times 4}$, which are sparsely sampled from a coarse-scale grid of boxes and replicated across time to form the initial proposals. An example of the 11 initial proposals used in our experiments is illustrated in Figure 5.2.

The initial proposals are then progressively updated to better classify and localize the actions. At each step s, we update the proposals by performing the following processes in order:

- **Extend**: the proposals are temporally extended to the adjacent clips to include longer-range temporal context, and the temporal extension is adaptive to the movement of actions, as described in Section 5.3.3.

- **Refine**: the extended proposals are forwarded to the spatial refinement, which outputs the classification and regression results, as presented in Section 5.3.2.

- **Update**: all proposals are updated using a simple greedy algorithm, i.e., each proposal is replaced by the regression output with the highest classification score:

$$b_i^s \doteq l_i^s(c^*), \quad c^* = \arg\max_c p_i^s(c), \tag{5.1}$$

Algorithm 1: STEP Action Detection for Clip I_t

Input : video clips $\mathcal{I}$, initial proposals B^0, and maximum steps S_{max}

Output: detection results $\left\{(p_i^{S_{max}}, l_i^{S_{max}})\right\}_{i=1}^{M}$

extract convolutional features for video clips $\mathcal{I}$
for $s \leftarrow 1$ **to** S_{max} **do**
    if $s == 1$ **then**
        // initial proposals
        $\tilde{B}^{s-1} \leftarrow B^0$
    else
        // temporal extension (Sec.5.3.3)
        $\tilde{B}^{s-1} \leftarrow \text{Extend}(B^{s-1})$
    end
    // spatial refinement (Sec.5.3.2)
    $\left\{(p_i^s, l_i^s)\right\}_{i=1}^{M} \leftarrow \text{Refine}(\tilde{B}^{s-1})$
    // update proposals (Eq.5.1)
    $B^s \leftarrow \text{Update}\left(\left\{(p_i^s, l_i^s)\right\}_{i=1}^{M}\right)$
end

where c is an action class, $p_i^s \in \mathcal{R}^{(C+1)}$ is the probability distribution of the ith proposal over C action classes plus background, $l_i^s \in \mathcal{R}^{K\times 4\times C}$ denotes its parameterized coordinates (for computing the localization loss in Eq. 5.3) at each frame for each class, and $\doteq$ indicates decoding the parameterized coordinates. We summarize the outline of our detection algorithm in Algorithm 1.

5.3.2 Spatial Refinement

At each step s, the spatial refinement solves a multi-task learning problem that involves action classification and localization regression. Accordingly, we design a two-branch architecture, which learns separate features for the two tasks, as illustrated in Figure 5.3. Our motivation is that the two tasks have substantially different objectives

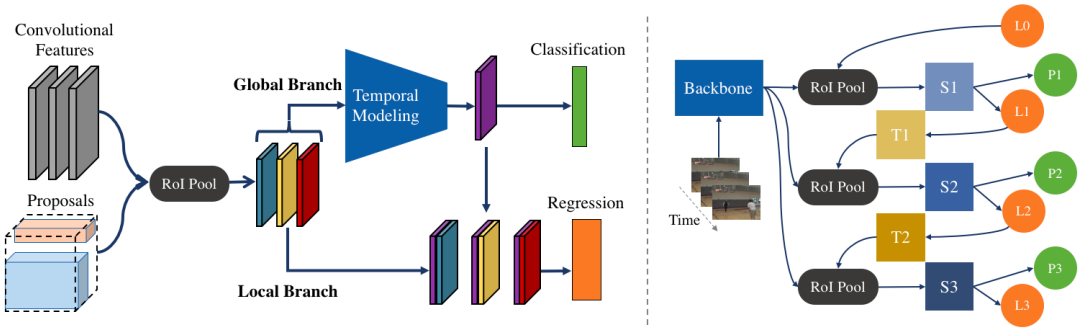


Figure 5.3: Left: the architecture of our two-branch network. Right: the illustration of our progressive learning framework, where "S" indicates spatial refinement, "T" temporal extension, "P" classification, and "L" localization, the numbers correspond to the steps, and "L0" denotes the initial proposals.

and require different types of information. For accurate action classification, it demands context features in both space and time, while for robust localization regression, it needs more precise spatial cues at frame level. As a result, our two-branch network consists of a *global branch* that performs spatio-temporal modeling on the entire input sequence for action classification, as well as a *local branch* that performs bounding box regression at each frame.

Given the frame-level convolutional features and the tubelet proposals for the current step, we first extract regional features through an ROI pooling [143]. Then we take the regional features to the global branch for spatio-temporal modeling and produce the global feature. Each global feature encodes the context information of a whole tubelet and is further used to predict the classification output p_i^s. Moreover, the global feature is concatenated with the corresponding regional features at each frame to form the local feature, which is used to generate the class-specific regression output l_i^s. Our local feature not only captures the spatio-temporal context of a tubelet but also extracts the local details of each frame. By jointly training the two branches, the network learns the two separate

features that are informative and adaptable for their own tasks.

Training Loss. We enforce a multi-task loss to jointly train for action classification and tubelet regression. Let $\mathcal{P}^s$ denote the set of selected positive samples and $\mathcal{N}^s$ the set of negative samples at step s (the sampling strategy is described in Section 5.3.4). We define the training loss $\mathcal{L}^s$ as:

$$\mathcal{L}^s = \sum_{i \in \{\mathcal{P}^s, \mathcal{N}^s\}} \mathcal{L}_{cls}(p_i^s, u_i) + \lambda \sum_{i \in \mathcal{P}^s} \mathcal{L}_{loc}(l_i^s(u_i), v_i), \tag{5.2}$$

where u_i and v_i are the ground truth class label and localization target for the ith sample, and λ is the weight to control the importance of the two loss terms. We employ the multi-class cross-entropy loss as the classification loss $\mathcal{L}_{cls}(p_i^s, u_i) = -\log p_i^s(u_i)$ in Eq. 5.2. We define the localization loss using the averaged $\ell_{1,\text{smooth}}$ between predicted and ground truth bounding boxes over the frames of a clip:

$$\mathcal{L}_{loc}(l_i^s(u_i), v_i) = \frac{1}{K} \sum_{k=1}^{K} \ell_{1,\text{smooth}}(l_{i,k}^s(u_i) - v_{i,k}). \tag{5.3}$$

We apply the same parameterization for $v_{i,k}$ as in [162] by using a scale-invariant center translation and a log-space height/width shift relative to the bounding box.

5.3.3 Temporal Extension

Video temporal information, especially the long-term temporal dependency, is critical for accurate action classification [9, 39]. In order to leverage longer range of temporal context, we extend the proposals to include in more frames as input. However, the

extension is not trivial since the spatial displacement problem becomes even more severe for longer sequences, as illustrated in Figure 5.1. Recently, some negative impacts caused by the spatial displacement problem for action detection have also been observed by [14, 145], which simply replicate 2D proposals across time to increase longer temporal length.

With the intention to alleviate the spatial displacement problem, we perform temporal extension *progressively* and *adaptively*. From the second step, we extend the tubelet proposals to the two adjacent clips at a time. In other words, at each step $1 \leq s < S_{max}$, the proposals B^s with length K^s are extended to $\tilde{B}^s = B^s_{-1} \circ B^s \circ B^s_{+1}$ with length $K^s + 2K$, where $\circ$ denotes concatenation. Additionally, the temporal extension is adaptive to action movement by taking advantage of the regressed tubelets from the previous step. We introduce two methods to enable the temporal extension to be adaptive as described in the following.

Extrapolation. By assuming that the spatial movement of an action satisfies a linear function approximately within a short temporal range, such as a 6-frame clip, we can extend the tubelet proposals by using a simple linear extrapolation function (also shown in Figure 5.4):

$$B^s_{+1,k} = B^s_{K^s} + \frac{k}{K-1}(B^s_{K^s} - B^s_{K^s-K+1}). \tag{5.4}$$

A similar function can be applied to B^s_{-1} to adapt to the movement trend, but the assumption would be violated for long sequences and therefore results in drifted estimations.

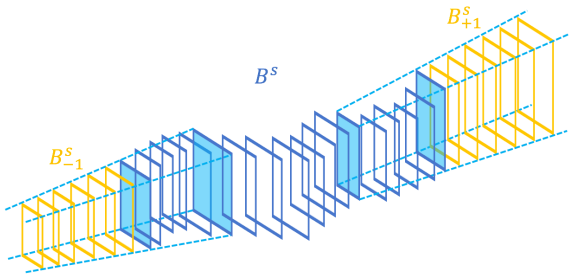


Figure 5.4: Illustration of extrapolation for adaptive temporal extension. Blue shaded boxes are the first and last bounding boxes of the corresponding tubelets.

Anticipation. We can also achieve the adaptive temporal extension by location anticipation, i.e., training an extra regression branch to conjecture the tubelet locations in adjacent clips based on the current clip. Intuitively, the anticipation requires the network to infer the movement trend in adjacent clips by action modeling in the current clip. A similar idea is explored in [153], where location anticipation is used at the region proposal stage.

We formulate our location anticipation as a residual learning problem [34, 163] based on the assumption that the tubelets of two adjacent clips differ from each other by a small residual. Let x indicate the features forwarded to the output layer f of the location regressor $L^s = f(x)$ at step s. So the anticipated locations can be obtained as:

$$L^s_{-1} = L^s + f_{-1}(x), \;\; L^s_{+1} = L^s + f_{+1}(x), \tag{5.5}$$

where f_{-1} and f_{+1} are the anticipation regressors, which are lightweight and introduce negligible computational overhead. L^s_{-1} and L^s_{+1} are then decoded to the proposals B^s_{-1} and B^s_{+1}. The loss function of location anticipation is defined in a similar way as Eq. 5.3, and combined with $\mathcal{L}_{cls}$ and $\mathcal{L}_{loc}$ with a coefficient γ to form the overall loss.

5.3.4 Network Training

Although STEP involves multiple progressive steps, the whole framework can be trained end-to-end to optimize the models at different steps jointly. Compared against the step-wised training scheme used in [160], our joint training is simpler to implement, runs more efficiently, and achieves better performance in our experiments.

Given a mini-batch of training data, we first perform an $(S_{max} - 1)$-step inference pass, as illustrated in the right of Figure 5.3, to obtain the inputs needed for all progressive steps. In practice, the detection outputs $\{(p_i^s, l_i^s)\}_{i=1}^M$ at each step are collected and used to select the positive and negative samples $\mathcal{P}^s$ and $\mathcal{N}^s$ for training. We accumulate the losses of all steps and back-propagate to update the whole model at the same time.

Coping with Distribution Change. Compared to the prior work that performs detection in one run, our training could be more challenging as the input/output distributions change over steps. As shown in Figure 5.5, the input distribution is right-skewed or centered in a low-IoU level at early steps, and reverses at later steps. This is because our approach starts from a coarse-scale grid (see Figure 5.2) and progressively refines them towards generating high-quality proposals. Accordingly, the range of output distribution (i.e., the scale of offset vectors) decreases over steps.

Inspired by [157], we tackle the distribution change in three ways. First, separate headers are used at different steps to adapt to the different input/output distributions. Second, we increase IoU thresholds over the multiple steps. Intuitively, a lower IoU threshold at early steps tolerates the initial proposals to include sufficient positive samples

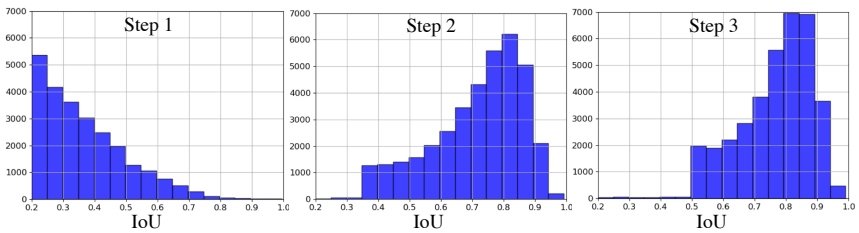


Figure 5.5: Change of input distribution (IoU between input proposals and ground truth) over steps on UCF101.

and a higher IoU threshold at late steps encourages high-quality detection. Third, a hard-aware sampling strategy is employed to select more informative samples during training.

Hard-Aware Sampling. We design the sampling strategy based on two principles: (i) the numbers of positive and negative samples should be roughly balanced, and (ii) the harder negatives should be selected more often. To measure the "hardness" of a negative sample, we use the classification scores from the previous step. The tubelet with a high confidence but a low overlap to any ground truth is viewed as a hard sample. We calculate the overlap of two tubelets by averaging the IoU of bounding boxes over K frames of the target clip. So the negative samples with higher classification scores will be sampled with a higher chance.

Formally, given a set of proposals and the overlap threshold τ^s at step s, we first assign positive labels to the candidates with the highest overlap with ground truth. This is to ensure that each ground truth tube has at least one positive sample. After that, the proposals having an overlap higher than τ^s with any ground truth tube are added to the positive pool and the rest to the negative pool. We then sample $|\mathcal{P}^s|$ positives and $|\mathcal{N}^s|$ negatives from the two pools, respectively, with the sampling probability proportional to the classification score. For the first step, the highest overlap with ground truth tubes is

used as the score for sampling. Each selected positive in $\mathcal{P}^s$ is assigned to the ground truth tube with which it has the highest overlap. Note that a single proposal can be assigned to only one ground truth tube.

5.3.5 Full Model

We can also integrate our model with the common practices for video action detection [14, 145, 151], such as two-stream fusion and tubelet linking.

Scene Context. It has been proven to be beneficial to object and action detection [122, 147]. Intuitively, some action-related semantic clues from scene context can be utilized to improve action classification, for example, the scene of a basketball court for recognizing "basketball dunk". We incorporate scene context by concatenating extended features to original regional features in the global branch. The extended features can be obtained by RoI pooling of the whole image. So the global features encode both spatial and temporal context useful for action classification.

Two-Stream Fusion. Most previous methods use late fusion to combine the results at test time, i.e., the detections are obtained independently from the two streams and then fused using either mean fusion [145] or union fusion [151]. In this work, we also investigate early fusion for two-stream fusion, which concatenates RGB frames and optical flow maps in channel and input to the network as a whole. Intuitively, early fusion can model the low-level interactions between the two modalities and also obviates the need for training two separate networks. In addition, a hybrid fusion can be further performed

to combine detection results from the early fusion and the two streams. Our experiment shows that early fusion outperforms late fusion, and hybrid fusion achieves the best performance.

Tubelet Linking. Given the clip-level detection results, we link them in space and time to construct the final action tubes. We follow the same linking algorithm as described in [145], apart from that we do not apply global non-maximum suppression across classes but perform temporal trimming over the linked paths as commonly used in [147, 150]. The temporal trimming enforces consecutive boxes to have smooth classification scores by solving an energy maximization problem via dynamic programming.

5.4 Experiments

In this section, we describe the experiments to evaluate STEP and compare against the recent competing algorithms. We start by performing a variety of ablation studies to better understand the contributions of each individual component in our approach. We then report comparisons to the state-of-the-art methods, provide in-depth analysis, and present the qualitative detection results.

5.4.1 Experimental Setup

Datasets. We evaluate our approach on the two benchmarks: UCF101 [69] and AVA [14]. In comparison with other action detection datasets, such as J-HMDB and UCFSports, the two benchmarks are much larger and more challenging, and more importantly, they are temporally untrimmed, which fits better to the spatio-temporal action detection task.

UCF101 is originally an action classification dataset collected from online videos, and a subset of 24 classes with 3,207 videos are provided with the spatio-temporal annotations for action detection. Following the standard evaluation protocol [145], we report results on the first split of the dataset. AVA contains complex actions and scenes sourced from movies. We use the version 2.1 of AVA, which consists of the annotations at 1 fps over 80 action classes. Following the standard setup in [14], we report results on the most frequent 60 classes that have at least 25 validation examples per class.

Evaluation Metrics. We report the frame-level mean average precision (frame-mAP) with an IoU threshold of 0.5 for both datasets. This metric allows us to evaluate the quality of the detection results independently of the linking algorithm. We also use the video-mAP on UCF101 to compare with the state-of-the-art results.

Implementation Details. For the experiments on UCF101, we use VGG16 [161] pre-trained on ImageNet [112] as the backbone network. Although more advanced models are available, we choose the same backbone as [145] for fair comparisons. The input of the backbone are video frames resized to 400×400 with the clip length $K = 6$. Similar to [145], we stack 5 consecutive optical flow maps as a whole for the optical flow input. Table 5.1 shows the details of our two-branch architecture. The network takes as inputs a sequence of $512 \times 25 \times 25$ feature maps from the backbone network (i.e., VGG16) as well as a set of proposal tubelets. For each proposal, an RoI pooling layer extracts a sequence of fixed-length regional features from the feature maps. For temporal modeling in the global branch, we first spatially extend each proposal tubelet to incorporate more scene

Layer		Output size
Global branch		
conv1	$3 \times 3 \times 3$, 1024	$6 \times 7 \times 7$
	max pool	
conv2	$3 \times 3 \times 3$, 512	$3 \times 7 \times 7$
	max pool	
conv3	$3 \times 3 \times 3$, 256	$1 \times 7 \times 7$
	average pool	
fc1(2)	4096	$1 \times 1 \times 1$
out	$C + 1$, softmax	$1 \times 1 \times 1$
Local branch		
fc1(2)	4096	$1 \times 1 \times 1$
out	$4 \times (C + 1)$	$1 \times 1 \times 1$

Table 5.1: Architecture of the two-branch network, where $T \times H \times W, N$ represent the dimensions of convolutional kernels and output feature maps.

context, as described in Section 5.3.5. We then forward the extended features to three 3D convolutional layers to obtain the global features. To perform action classification, the global features are flatten and fed into a sequence of fully connected (`fc`) layers, which finally output the softmax probability estimates over C classes plus background. To perform tubelet regression, the global features are concatenated along channel dimension with the regional features at each frame and then fed into another sequence of `fc` layers, which produce a class-specific regression output with the shape $4 \times (C + 1)$ for each frame. We train our models for 35 epochs using Adam [164] with a batch size of 4. We set the initial learning rate to 5×10^{-5} and perform step decay after 20 and 30 epochs with the decay rate 0.1.

For the experiments on AVA, we adopt I3D [9] (up to `Mixed_4f`) pre-trained on Kinetics-400 [88] as the backbone network. Instead of introducing extra 3D convolutional layers for temporal modeling as in Table 5.1, we take the two layers `Mixed_5b` and

`Mixed_5c` of I3D for temporal modeling in our global branch. Input frames are sampled in 12 fps and resized to 400×400, and the clip length is set to $K = 12$. We use 34 initial proposals and perform temporal extension only at the third step. The initial proposals are generated following the practice in [160]. In details, we generate the initial proposals using a two-level spatial pyramid with $[4/3,\ 2]$ scales and $[5/6,\ 3/4]$ overlap for each spatial scale. In other words, a sliding window with size $3W/4 \times 3H/4$ pixels and overlap ratio $5/6$ is used for the first level, and a sliding window with size $W/2 \times H/2$ pixels and overlap ratio $3/4$ is used for the second level. Here, W and H denote the width and height of the frames, respectively. As the classification is more challenging on AVA, we first pre-train our model for an action classification task using the spatial ground truth of training set. We then train the model for action detection with a batch size of 4 for 10 epochs. We do not use optical flow on this dataset due to the heavy computation and instead combine results of two RGB models. Our initial learning rate is 5×10^{-6} for the backbone network and 5×10^{-5} for the two-branch networks, and step decay is performed after 6 epochs with the decay rate 0.1.

For all experiments, we extract optical flow (if used) with Brox [165], and perform data augmentation to the whole sequence of frames during training, including random flipping and cropping.

5.4.2 Ablation Study

We perform various ablation experiments on UCF101 to evaluate the impacts of different design choices in our framework. For all experiments in this section, we employ

S_{max}	s			
	1	2	3	4
1	51.5	-	-	-
2	56.6	60.7	-	-
3	57.1	61.8	62.6	-
4	**58.2**	**62.1**	**62.8**	**62.7**

Mode	f-mAP
RGB	66.7
Flow	63.5
Late	70.7
Early	74.3
Hybrid	**75.0**

Table 5.2: Comparisons of frame-mAP (%) of our models trained with different numbers of steps (left), and different input modalities and fusion methods (right).

the 11 initial proposals as shown in Figure 5.2 and RGB only, unless explicitly mentioned otherwise, and frame-mAP is used as the evaluation metric.

Effectiveness of Spatial Refinement. Our primary design of STEP is to progressively tackle the action detection problem through a few steps. We thus first verify the effectiveness of progressive learning by comparing the detection results at different steps with the spatial refinement. No temporal extension is applied in this comparison. Table 5.2 (left) demonstrates the step-wise performance under different maximum steps S_{max}. Since our approach starts from the coarse-scale proposals, performing spatial refinement once is insufficient to achieve good results. We observe that the second step improves results consistently and substantially, indicating that the updated proposals have higher quality and provide more precise information for classification and localization. Further improvement can be obtained by additional steps, suggesting the effectiveness of our progressive spatial refinement. We use 3 steps for most of our experiments as the performance saturates after that. Note that using more steps also improves the results of early steps, due to the benefits of our multi-step joint training.

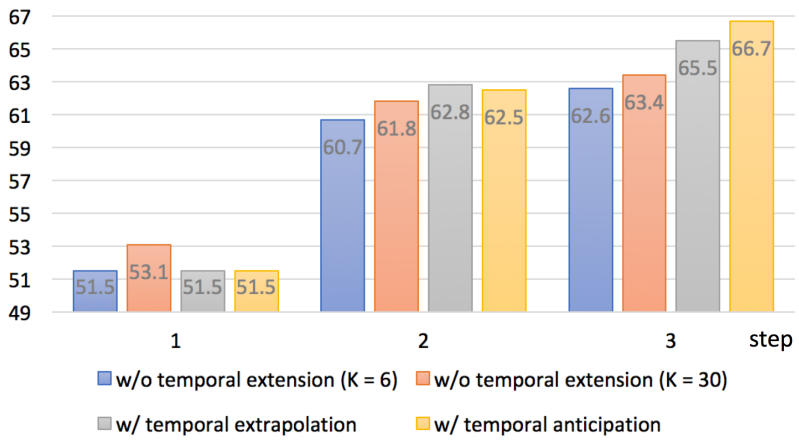


Figure 5.6: Comparison of frame-mAP (%) of our models trained with and without temporal extension.

Effectiveness of Temporal Extension. In addition to the spatial refinement, our progressive learning contains the temporal extension to progressively process a longer sequence at each step. We compare the detection results with and without temporal extension in Figure 5.6. We show the results of the models taking $K = 6$ and $K = 30$ frames as inputs directly without temporal extension, and the results of the extrapolation and anticipation methods. Note that the models with temporal extension also deal with 30 frames at the third step (extension process: $6 \rightarrow 18 \rightarrow 30$).

Both of the temporal extension methods outperform the baseline ($K = 6$) by a large margin, which clearly shows the benefit of incorporating longer-range temporal context for action classification. More remarkably, simply taking $K = 30$ frames as input without temporal extension results in inferior performance, validating the importance of adaptively extending the temporal scale in the progressive manner. Furthermore, we observe that anticipation performs better than extrapolation for longer sequences, indicating that anticipation can better capture nonlinear movement trends and therefore generate better extensions.

Fusion Comparison. Table 5.2 (right) presents the detection results of different fusions: late, early and hybrid fusion. In all cases, using both modalities improves the performance compared to individual ones. We find that early fusion outperforms late fusion, and attribute the improvement to modeling between the two modalities at the early stage. Hybrid fusion achieves the best result by further utilizing the complementary information of different methods.

Miscellaneous. We describe several techniques to improve the training in Section 5.3, including incorporating scene context, hard-award sampling and increasing IoU threshold. To validate the contributions of the three techniques, we conduct ablation experiments by removing one at a time, which correspondingly results in a performance drop of 2.5%, 1.5% and 1%. In addition, we observe that incorporating scene context provides more gains for later steps, suggesting that scene context is more important for action classification when bounding boxes become tight.

5.4.3 Runtime Analysis

Although STEP involves a multi-step optimization, our model is efficient since we only process a small number of proposals. STEP runs at 21 fps using early fusion with 11 initial proposals and 3 steps on a single GPU, which is comparable with the clip based approach (23 fps) [145] and much faster than the frame based method (4 fps) [149]. Figure 5.7(a) demonstrates the speeds of our approach with increasing number of steps under the settings with and without temporal extension. We also report the running time and detection performance of our approach (w/o temporal extension for 3 steps) with

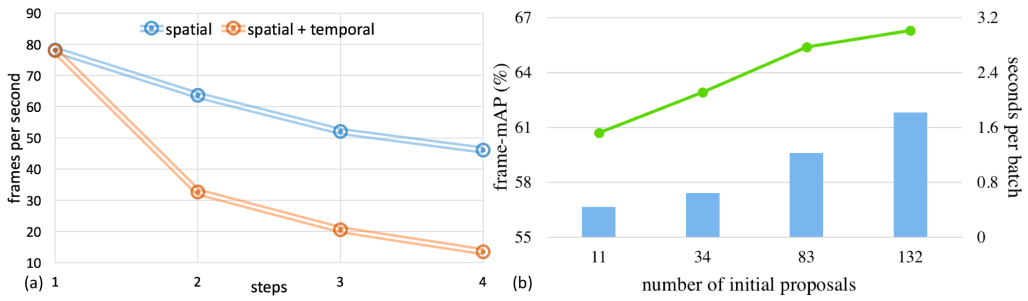


Figure 5.7: Analysis of runtime of our approach under various settings: (a) the inference speeds using different step numbers with and without temporal extension, and (b) the detection results (green dots) and speeds (blue bars) using different numbers of initial proposals.

increasing number of initial proposals in Figure 5.7(b). We observe substantial gains in detection accuracy by increasing the number of initial proposals, but it also results in slowed inference speed. This trade-off between accuracy and speed can be controlled according to a specified time budget.

5.4.4 Comparison with State-of-the-Art Results

We compare our approach with the state-of-the-art methods on UCF101 and AVA in Tables 5.3 and 5.4. Following the standard settings, we report the frame-mAP at IoU threshold 0.5 on both datasets and the video-mAP at various IoU thresholds on UCF101. STEP consistently performs better than the state-of-the-art methods on UCF101, and brings a clear gain in frame-mAP, producing 5.5% improvement over the second best result. Our approach also achieves superior result on AVA, outperforming the recently proposed ACRN by 1.2%. Notably, STEP performs detection simply from a handful of initial proposals, while other competing algorithms rely on a great amount of densely sampled anchors or an extra person detector trained with external large-scale image ob-

Method	frame-mAP	video-mAP		
	0.5	0.05	0.1	0.2
MR-TS [149]	65.7	78.8	77.3	72.9
ROAD [151]	-	-	-	73.5
CPLA [153]	-	79.0	77.3	73.5
RTPR [147]	-	81.5	80.7	76.3
PntMatch [154]	67.0	79.4	77.7	76.2
T-CNN [146]	67.3	78.2	77.9	73.1
ACT [145]	69.5	-	-	76.5
Ours	**75.0**	**84.6**	**83.1**	**76.6**

Table 5.3: Comparison with the state-of-the-art methods on UCF101 by frame-mAP (%) and video-mAP (%) under different IoU thresholds.

Method	Flow	frame-mAP
Single Frame [14]	✓	14.2
I3D [14]		14.7
I3D [14]	✓	15.6
ACRN [122]	✓	17.4
Ours		**20.7**

Table 5.4: Comparison with the state-of-the-art methods on AVA by frame-mAP (%) under IoU $= 0.5$. "*" means the results obtained by incorporating optical flow.

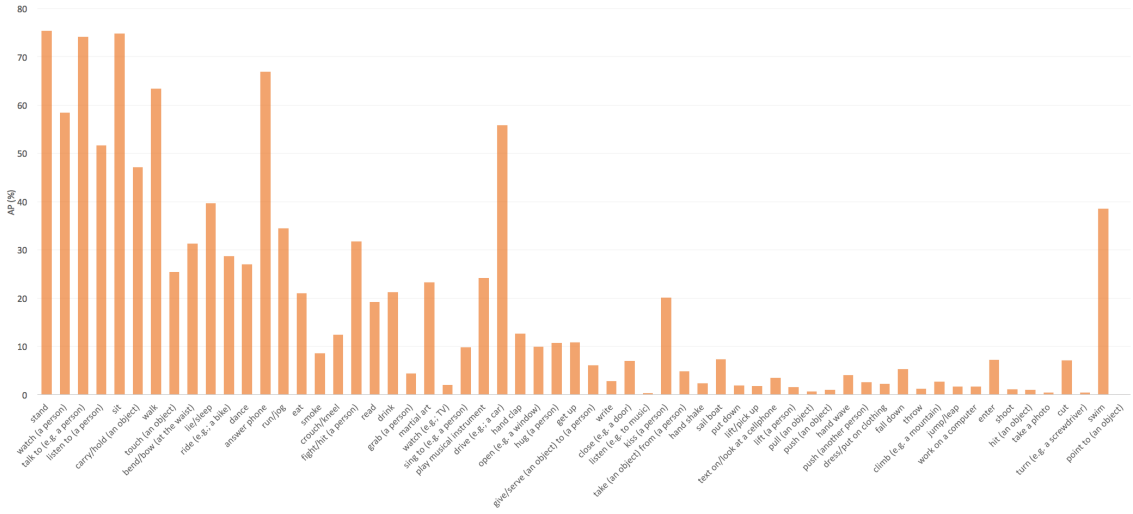


Figure 5.8: Comparison of the per-class breakdown frame-AP at IoU threshold 0.5 on AVA.

ject detection datasets. We also demonstrates the per-class breakdown frame-AP on AVA in Figure 5.8.

5.4.5 Qualitative Results

We first show some examples to illustrate the spatial displacement problem in Figure 5.9. We further analyze the spatial displacement problem on UCF101 by calculating the minimum IoU within tubes (MIUT). Given a ground truth action tube, MIUT is de-

Figure 5.9: Examples of the spatial displacement problem. Red boxes indicate the ground truth bounding boxes and blue ones the spatial grids. From top to bottom are LongJump (ID: 12), FloorGymnastics (ID: 8) and CliffDiving (ID: 4).

Figure 5.10: MIUT of ground truth action tubes on UCF101. K denotes different tube lengths, and red dash line corresponds to MIUT $= 0.5$.

fined by the minimum IoU overlap between the center bounding box (i.e., the box of the center frame) and the other bounding boxes within the tube. Figure 5.10 demonstrates the statistics of different actions with different length using ground truth action tubes in the validation set. We observe that the spatial displacement problem is not very obvious for short clips (e.g., $K = 6$), as most action classes have high MIUT values. However, the spatial displacement problem becomes more severe for most actions when the sequence length increases. For example, "Skijet" (ID: 18) has a 0.12 MIUT and "CliffDiving"

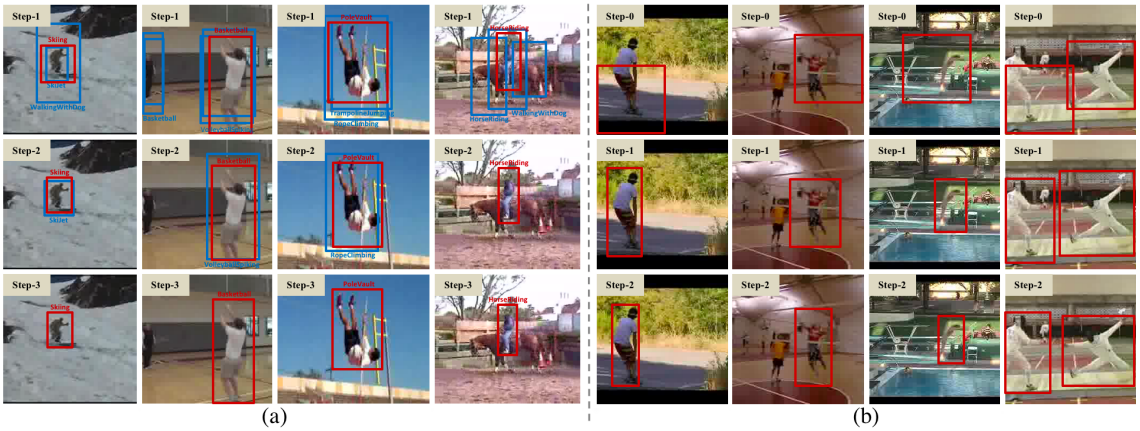


Figure 5.11: Examples of the detection results on UCF101. Red boxes indicate correct detection and blue ones misclassification. (a) illustrates the effect of progressive learning to improve action classification over steps. (b) demonstrates the regression outputs by spatial refinement at each step.

(ID: 4) has a 0.17 MIUT when $K = 30$, indicating both actions encounter large spatial displacements within the tubes.

We then visualize the detection results of our approach at different steps in Figure 5.11. Each row indicates the detection outputs at a certain step. A bounding box is labeled in red if the detection result is correct, otherwise it is labeled in blue. Figure 5.11(a) demonstrates the effect of progressive learning for more accurate action classification. It can be observed by the fact that the blue boxes are eliminated at later steps. In Figure 5.11(b), the first row corresponds to the initial proposals and the next two rows show the effect of spatial refinement of the proposals over steps. It is clear that the proposals progressively move towards the persons performing the actions and better localization results are obtained at later steps. Although starting from coarse-scale proposals, our approach is robust to various action scales thanks to the progressive spatial refinement, as illustrated in Figure 5.12.

Figure 5.12: Examples of the small scale action detection by our approach. Red boxes indicate the initial proposals and orange ones the detection outputs.

5.5 Conclusion

In this chapter, we have presented the spatio-temporal progressive learning framework STEP for video action detection. STEP involves spatial refinement and temporal extension, where the former starts from sparse initial proposals and iteratively updates bounding boxes, and the latter gradually and adaptively increases sequence length to incorporate more related temporal context. STEP is found to be able to more effectively make use of longer temporal information by handling the spatial displacement problem in action tubes. Extensive experiments on two benchmarks show that STEP consistently brings performance gains by using only a handful of proposals and a few updating steps.

Chapter 6: Conclusion and Future Directions

In this dissertation, we study the temporal modeling of video data for action recognition and detection. In particular, we focus on extracting motion representation from raw video frames and modeling long-range temporal dependencies in full-length videos. We first presented a hierarchical contrastive learning framework that extracts lower-level and high-level motion features in a fully self-supervised manner. Next, we investigated the self-attention mechanism for temporal modeling and proposed a global temporal attention (GTA) module that significantly outperforms the standard non-local block for action recognition. We then presented collaborative memory (CM), an end-to-end video-level learning framework that is capable of capturing temporal dependencies beyond short clips. Finally, we introduced the spatio-temporal progressive learning framework (STEP) for video action detection, which is able to more effectively make use of longer temporal information by handling the spatial displacement problem in action tubes.

While the performance of video action recognition system has been significantly improved, there are still many remaining challenges for video understanding:

Reasoning about spatio-temporal relationships. Existing action detection methods simply view co-occurred actions as irrelevant categories and train independent classifiers for each class. However, the correlation and interaction of human actions, both in

space and time, provide rich context for actions and a wealth of commonsense information about the physical world. For example, "falling down" is likely to be followed by "lying", while "dancing" and "fighting" are very unlikely to co-occur (though they may share similar appearance and motion). We believe that explicit higher level reasoning with scenes, poses and objects will help develop more accurate and explainable action understanding models.

Better understanding of how motion information help action Recognition. Most of the current state-of-the-art action recognition models utilize optical flow as motion representation. However, it is still open questions why optical flow is helpful, what makes a flow method good for action recognition, and how we can make it better. Recent work [166] has shown that much of the value of optical flow, when used as input of the two-stream architectures, is that it is a representation invariant to appearance. This motivates us to explore more effective ways to utilize optical flow for action recognition. For example, it is appealing to design new models that can directly leverage the directional information encoded in optical flow, rather than simply taking optical flow a "black box" input.

Bibliography

[1] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

[2] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM MM*, 2007.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[4] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NeurIPS*, 2014.

[5] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

[6] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[7] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

[8] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017.

[9] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *CVPR*, 2017.

[10] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[11] Ji Lin, Chuang Gan, and Song Han. TSM: Temporal shift module for efficient video understanding. In *ICCV*, 2019.

[12] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019.

[13] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime TV-L1 optical flow. In *Joint Pattern Recognition Symposium*, 2007.

[14] Chunhui Gu, Chen Sun, Sudheendra Vijayanarasimhan, Caroline Pantofaru, David A Ross, George Toderici, Yeqing Li, Susanna Ricco, Rahul Sukthankar, and Cordelia Schmid. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *CVPR*, 2018.

[15] Xitong Yang, Xiaodong Yang, Ming-Yu Liu, Fanyi Xiao, Larry Davis, and Jan Kautz. STEP: Spatio-temporal progressive learning for video action detection. In *CVPR*, 2019.

[16] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *ICCV*, 2017.

[17] Zhichao Lu, Vivek Rathod, Ronny Votel, and Jonathan Huang. RetinaTrack: Online single stage joint detection and tracking. In *CVPR*, 2020.

[18] Yue-Hei Ng, Jonghyun Choi, Jan Neumann, and Larry Davis. ActionFlowNet: Learning motion representation for action recognition. In *WACV*, 2018.

[19] Heng Wang, Du Tran, Lorenzo Torresani, and Matt Feiszli. Video modeling with correlation networks. In *CVPR*, 2020.

[20] Jonathan C Stroud, David A Ross, Chen Sun, Jia Deng, and Rahul Sukthankar. D3D: Distilled 3D networks for video action recognition. In *WACV*, 2020.

[21] Laura Sevilla-Lara, Shengxin Zha, Zhicheng Yan, Vedanuj Goswami, Matt Feiszli, and Lorenzo Torresani. Only time can tell: Discovering temporal data for temporal modeling. *WACV*, 2021.

[22] Yi Zhu, Zhenzhong Lan, Shawn Newsam, and Alexander Hauptmann. Hidden two-stream convolutional networks for action recognition. In *ACCV*, 2018.

[23] Martin A Giese and Tomaso Poggio. Neural mechanisms for the recognition of biological movements. *Nature Reviews Neuroscience*, 2003.

[24] Johannes Bill, Hrag Pailian, Samuel J Gershman, and Jan Drugowitsch. Hierarchical structure is employed by humans during visual motion perception. *Journal of Vision*, 2019.

[25] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

[26] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

[27] Aaron Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

[28] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[29] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *ICCV Workshop*, 2019.

[30] J Yu Jason, Adam W Harley, and Konstantinos G Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *ECCV*, 2016.

[31] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha. Unsupervised deep learning for optical flow estimation. In *AAAI*, 2017.

[32] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016.

[33] Ali Diba, Vivek Sharma, Luc Van Gool, and Rainer Stiefelhagen. DynamoNet: Dynamic action and motion network. In *ICCV*, 2019.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[35] Behrooz Mahasseni, Xiaodong Yang, Pavlo Molchanov, and Jan Kautz. Budget-aware activity detection with a recurrent policy network. In *BMVC*, 2018.

[36] Xiaodong Yang, Pavlo Molchanov, and Jan Kautz. Multilayer and multimodal fusion of deep neural networks for video classification. In *ACM MM*, 2016.

[37] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet? In *CVPR*, 2018.

[38] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, 2018.

[39] Xiaodong Yang, Pavlo Molchanov, and Jan Kautz. Making convolutional networks recurrent for visual sequence learning. In *CVPR*, 2018.

[40] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *ICCV*, 2019.

[41] AJ Piergiovanni and Michael S Ryoo. Representation flow for action recognition. In *CVPR*, 2019.

[42] Lijie Fan, Wenbing Huang, Chuang Gan, Stefano Ermon, Boqing Gong, and Junzhou Huang. End-to-end learning of motion representation for video understanding. In *CVPR*, 2018.

[43] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[44] Heeseung Kwon, Manjin Kim, Suha Kwak, and Minsu Cho. Motionsqueeze: Neural motion feature learning for video understanding. In *ECCV*, 2020.

[45] Gucan Long, Laurent Kneip, Jose M Alvarez, Hongdong Li, Xiaohu Zhang, and Qifeng Yu. Learning image matching by simply watching video. In *ECCV*, 2016.

[46] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *CVPR*, 2018.

[47] Joel Janai, Fatma Guney, Anurag Ranjan, Michael Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *ECCV*, 2018.

[48] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016.

[49] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *ICCV*, 2017.

[50] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *CVPR*, 2017.

[51] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015.

[52] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *ECCV*, 2016.

[53] Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3D: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks. In *ECCV*, 2016.

[54] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *ICLR*, 2017.

[55] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. *NeurIPS*, 2020.

[56] Guillaume Lorre, Jaonary Rabarisoa, Astrid Orcesi, Samia Ainouz, and Stephane Canu. Temporal contrastive pretraining for video action recognition. In *WACV*, 2020.

[57] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. *CVPR*, 2021.

[58] Li Tao, Xueting Wang, and Toshihiko Yamasaki. Self-supervised video representation learning using inter-intra contrastive framework. In *ACM MM*, 2020.

[59] Ceyuan Yang, Yinghao Xu, Bo Dai, and Bolei Zhou. Video representation learning with visual tempo consistency. *arXiv preprint arXiv:2006.15489*, 2020.

[60] Asmaa Hosni, Christoph Rhemann, Michael Bleyer, Carsten Rother, and Margrit Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *PAMI*, 2012.

[61] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of CNNs for optical flow estimation. *PAMI*, 2019.

[62] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018.

[63] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[64] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

[65] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

[66] Zhang Zhang and Dacheng Tao. Slow feature analysis for human action recognition. *PAMI*, 2012.

[67] Deqing Sun, Stefan Roth, and Michael J Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *IJCV*, 2014.

[68] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, and Moritz Mueller-Freitag. The "something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017.

[69] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1212.0402*, 2012.

[70] Zhaofan Qiu, Ting Yao, Chong-Wah Ngo, Xinmei Tian, and Tao Mei. Learning spatio-temporal representation with local and global diffusion. In *CVPR*, 2019.

[71] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018.

[72] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[73] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv:1706.02677*, 2017.

[74] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[75] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *ECCV*, 2018.

[76] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *ECCV*, 2018.

[77] Yue Zhao, Yuanjun Xiong, and Dahua Lin. Recognize actions by disentangling components of dynamics. In *CVPR*, 2018.

[78] Nieves Crasto, Philippe Weinzaepfel, Karteek Alahari, and Cordelia Schmid. MARS: Motion-augmented RGB stream for action recognition. In *CVPR*, 2019.

[79] Dahun Kim, Donghyeon Cho, and Inso Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *AAAI*, 2019.

[80] Ce Liu. *Beyond pixels: Exploring new representations and applications for motion analysis*. PhD thesis, MIT, 2009.

[81] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-CAM++: Improved visual explanations for deep convolutional networks. *WACV*, 2018.

[82] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[83] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[84] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *NeurIPS*, 2015.

[85] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[86] Jie Hu, L Longfei Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi. Gather-excite: Exploiting feature context in convnets. In *NeurIPS*, 2018.

[87] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *ICLR*, 2020.

[88] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, and Paul Natsev. The Kinetics human action video dataset. *arXiv:1705.06950*, 2017.

[89] Kaiyu Yue, Ming Sun, Yuchen Yuan, Feng Zhou, Errui Ding, and Fuxin Xu. Compact generalized non-local network. In *NeurIPS*, 2018.

[90] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *ICCVW*, 2019.

[91] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *CVPR*, 2019.

[92] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[93] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *TPAMI*, 2012.

[94] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[95] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Timeception for complex action recognition. In *CVPR*, 2019.

[96] Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *CVPR*, 2020.

[97] Thomas M Strat and Martin A Fischler. Context-based vision: recognizing objects using information from both 2d and 3d imagery. *PAMI*, 1991.

[98] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *CVPR*, 2005.

[99] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[100] Saurabh Gupta, Bharath Hariharan, and Jitendra Malik. Exploring person context and local scene context for object detection. *arXiv preprint arXiv:1511.08177*, 2015.

[101] Geremy Heitz and Daphne Koller. Learning spatial context: Using stuff to find things. In *ECCV*, 2008.

[102] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014.

[103] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[104] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *CVPR*, 2018.

[105] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *CVPR*, 2018.

[106] Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Timothy P. Lillicrap. A simple neural network module for relational reasoning. In *NeurIPS*, 2017.

[107] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, 2019.

[108] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019.

[109] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[110] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *AAAI*, 2017.

[111] Chun-Fu Chen, Rameswar Panda, Kandan Ramakrishnan, Rogerio Feris, John Cohn, Aude Oliva, and Quanfu Fan. Deep analysis of cnn-based spatio-temporal representations for action recognition. *arXiv preprint arXiv:2010.11757*, 2020.

[112] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[113] Quanfu Fan, Chun-Fu Richard Chen, Hilde Kuehne, Marco Pistoia, and David Cox. More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation. In *NeurIPS*, 2019.

[114] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *ECCV*, 2018.

[115] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39. Springer, 2011.

[116] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015.

[117] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017.

[118] Christoph Feichtenhofer. X3D: Expanding architectures for efficient video recognition. In *CVPR*, 2020.

[119] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. *ICCV*, 2019.

[120] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *CVPR*, 2019.

[121] Linchao Zhu, Laura Sevilla-Lara, Du Tran, Matt Feiszli, Yi Yang, and Heng Wang. FASTER recurrent networks for efficient video classification. In *AAAI*, 2020.

[122] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *ECCV*, 2018.

[123] Jiajun Tang, Jin Xia, Xinzhi Mu, Bo Pang, and Cewu Lu. Asynchronous interaction aggregation for action detection. In *ECCV*, 2020.

[124] A. J. Piergiovanni, Anelia Angelova, and Michael S. Ryoo. Tiny video networks. *arxiv:1910.06961*, 2019.

[125] Okan Köpüklü, Neslihan Kose, Ahmet Gunduz, and Gerhard Rigoll. Resource efficient 3d convolutional neural networks. In *ICCV Workshop*, 2019.

[126] Z. Wu, H. Li, C. Xiong, Y. G. Jiang, and L. S. Davis. A dynamic frame selection framework for fast video recognition. *TPAMI*, 2020.

[127] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021.

[128] Bruno Korbar, Du Tran, and Lorenzo Torresani. SCSampler: Sampling salient clips from video for efficient action recognition. In *ICCV*, 2019.

[129] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[130] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *NeurIPS*, 2015.

[131] Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*, 2016.

[132] Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *NeurIPS*, 2016.

[133] Seil Na, Sangho Lee, Jisung Kim, and Gunhee Kim. A read-write memory network for movie story understanding. In *ICCV*, 2017.

[134] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 1982.

[135] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with context gating for video classification, 2017.

[136] Haoqi Fan, Yanghao Li, Bo Xiong, Wan-Yen Lo, and Christoph Feichtenhofer. Pyslowfast. `https://github.com/facebookresearch/slowfast`, 2020.

[137] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016.

[138] Fanyi Xiao, Yong Jae Lee, Kristen Grauman, Jitendra Malik, and Christoph Feichtenhofer. Audiovisual slowfast networks for video recognition. *arXiv preprint arXiv:2001.08740*, 2020.

[139] Chenxu Luo and Alan Yuille. Grouped spatial-temporal aggretation for efficient action recognition. In *ICCV*, 2019.

[140] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019.

[141] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *CVPR*, 2018.

[142] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *CVPR*, 2019.

[143] Ross Girshick. Fast R-CNN. In *ICCV*, 2015.

[144] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[145] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, and Cordelia Schmid. Action tubelet detector for spatio-temporal action localization. In *ICCV*, 2017.

[146] Rui Hou, Chen Chen, and Mubarak Shah. Tube convolutional neural network (T-CNN) for action detection in videos. In *ICCV*, 2017.

[147] Dong Li, Zhaofan Qiu, Qi Dai, Ting Yao, and Tao Mei. Recurrent tubelet proposal and recognition networks for action detection. In *ECCV*, 2018.

[148] Georgia Gkioxari and Jitendra Malik. Finding action tubes. In *CVPR*, 2015.

[149] Xiaojiang Peng and Cordelia Schmid. Multi-region two-stream R-CNN for action detection. In *ECCV*, 2016.

[150] Suman Saha, Gurkirt Singh, Michael Sapienza, Philip Torr, and Fabio Cuzzolin. Deep learning for detecting multiple space-time action tubes in videos. In *BMVC*, 2016.

[151] Gurkirt Singh, Suman Saha, Michael Sapienza, Philip Torr, and Fabio Cuzzolin. Online real-time multiple spatiotemporal action localisation and prediction. In *ICCV*, 2017.

[152] Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Learning to track for spatio-temporal action localization. In *ICCV*, 2015.

[153] Zhenheng Yang, Jiyang Gao, and Ram Nevatia. Spatio-temporal action detection with cascade proposal and location anticipation. In *BMVC*, 2017.

[154] Yuancheng Ye, Xiaodong Yang, and Yingli Tian. Discovering spatio-temporal action tubes. *JVCI*, 2019.

[155] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *CVPR*, 2016.

[156] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. DRAW: A recurrent neural network for image generation. In *ICML*, 2015.

[157] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *CVPR*, 2018.

[158] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region and semantic segmentation-aware CNN model. In *ICCV*, 2015.

[159] Spyros Gidaris and Nikos Komodakis. Attend refine repeat: Active box proposal generation via in-out localization. In *BMVC*, 2016.

[160] Mahyar Najibi, Mohammad Rastegari, and Larry Davis. G-CNN: An iterative grid based object detector. In *CVPR*, 2016.

[161] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[162] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[163] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael Jordan. Unsupervised domain adaptation with residual transfer networks. In *NeurIPS*, 2016.

[164] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[165] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004.

[166] Laura Sevilla-Lara, Yiyi Liao, Fatma Güney, Varun Jampani, Andreas Geiger, and Michael J Black. On the integration of optical flow and action recognition. In *German Conference on Pattern Recognition*, pages 281–297. Springer, 2018.