Spatio-Temporal Modeling for Action Recognition in Videos

Guoxi Huang

Doctor of Philosophy

University of York
Computer Science

August 2022

Abstract

Technological innovation in the field of video action recognition drives the development of video-based real-world applications. This PhD thesis provides a new set of machine learning algorithms for processing videos efficiently, leading to outstanding results in human action recognition in videos. First of all, two video representation extraction methods, Temporal Squeezed Pooling (TSP) and Pixel-Wise Temporal Projection (PWTP), are proposed in order to enhance the discriminative video feature learning abilities of Deep Neural Networks (DNNs). TSP enables spatio-temporal modeling by temporally aggregating the information from long video frame sequences. PWTP is an improved version TSP, which filters out static appearance while performing information aggregation. Secondly, we discuss how to address the long-term dependency modeling problem of video DNNs. To this end, we develop two spatio-temporal attention mechanisms, Region-based Non-local (RNL) and Convolution Pyramid Attention (CPA). We devise an attention chain by connecting the RNL or CPA module to the Squeeze-Excitation (SE) operation. We demonstrate how the attention mechanisms can be embedded into deep networks to alleviate the optimization difficulty. Finally, we are focused on tackling the problem of heavy computational cost in video models. To this end, we introduce the concept of busy-quiet video disentangling for exceedingly fast video modeling. We propose the Motion Band-Pass Module (MBPM) embedded into the Busy-Quiet Net (BQN) architecture to reduce videos' information redundancy in the spatial and temporal dimensions. The BQN architecture is extremely lightweight while still performing better than other heavier models. Extensive experiments for all the proposed methods are provided on multiple video benchmarks, including UCF101, HMDB51, Kinetics400 and Something-Something V1.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my appreciation and gratitude to the people who physically, mentally, academically and domestically support my PhD journey.

First, I am sincerely grateful to my supervisor Dr. Adrian G. Bors, for all his priceless and innovative advice on my research and career. He gave me great support no matter if I encountered difficulties in academics or in life. Whenever my research is trapped into a bottleneck, his resourceful knowledge and endless ideas got me back on track. My gratitude also goes to my assessor Pro. Richard Wilson for the helpful discussions in the TAP meetings and his precise assessment of my work.

My friends and peers working around me teach and inspire me a lot. Ye Yu, Fei Ye, Zeynel Abidin, - I would like to express my special thanks to them for forging a wonderful working and researching environment.

Lastly and most importantly, I would like to thank my family. My very special gratitude goes to my father Fangjun Huang, my mother Wenjie Lin, my older sister Guorong Huang, my younger sister Zerong Huang and my fiancee Leo Russell Willis. I feel grateful for their unconditional support and encouragement. During the four years of my PhD career, Leo has been devoting great care to me. My PhD journey would be more arduous and less fun without her company.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References. Some parts of this thesis have been published in conference proceedings and journals; where items were published jointly the author of this thesis is responsible for the material presented here. For each published item the primary author is the first listed author.

- Guoxi Huang and Adrian G. Bors. Learning spatio-temporal representations with temporal squeeze pooling. Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020
- Guoxi Huang and Adrian G. Bors. Region-based non-local operation for video classification. Proceedings of IEEE International Conference on Pattern Recognition (ICPR), 2021
- Guoxi Huang and Adrian G. Bors. Busy-Quiet Video Disentangling for Video Classification. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2022
- Guoxi Huang and Adrian G. Bors. BQN: Busy-Quiet Net Enabled by Motion Band-Pass Module for Action Recognition. IEEE Transactions on Image Processing.

Guoxi Huang
August 2022

Chapter 1

Introduction

A video is a recording of moving images, with or without audio, usually saved as a digital file, DVD *et al*. AI video recognition is an area of computer vision and pattern recognition that trains digital electronic machines to interpret and identify the motion patterns in videos by using machine learning algorithms. In the principles of computer vision, machine learning for video action recognition deals with video data processing and video analysis, to achieve an understanding of digital video information. The main theme of this PhD thesis is to design machine learning algorithms to address practical problems encountered in video action recognition tasks.

1.1 Motivation

Video action recognition is a fundamental problem in video understanding, having many real-world applications, including autonomous driving technology, security, defense, safety, controlling drones, robots, human-computer interaction and video gaming. Over the last decade, video action recognition has attracted increasing research interests in computer vision and has been through fast development with the rise of deep learning [77] and the availability of large-scale labeled video datasets [114, 73, 70, 48]. The downstream tasks such as video retrieval and action localization also benefit from advances in video action recognition technology.

A video is essentially a sequence of RGB images showing the evolution of objects and scenes along the temporal dimension. Hence, the video recognition problem can be considered to be an extension of image recognition from space to space-time. However, video recognition cannot be naively treated as an image recognition problem, because the most important factor for video understanding is the motion clues invisibly existing in the temporal dimension. The

early work [69] attempted to distinguish different actions in videos by only processing one frame for each video. The unsatisfactory performance of this method seems to be taken for granted now, as this single-frame method fails to capture motion clues between video frames. Therefore, how to extract motion clues from videos in an efficient and effective way becomes the core of video recognition, waiting for us to address in this thesis.

Video processing inherits most of the challenges from image processing, including the influence of illumination variation, occlusion between various moving objects, resolution scale, noise and changes caused by environmental factors (rain, snow, smoke) or compression artifacts, perspective projection effect and so on. Meanwhile, video recognition methods need to deal with the intrinsic problems of video data, such as various movement speeds, viewpoint changes across frames and camera movement. With the emergence of deep learning [77], the main research direction in video action recognition has been shifted from constructing hand-crated representations to building efficient spatio-temporal networks. The family of deep neural networks (DNNs) is a class of machine learning algorithms inspired by the biological neural networks that constitute brains. Due to their high non-linearity, DNNs are capable of learning high-level semantic features, demonstrating leading-edge advantages over traditional hand-crafted features in many machine learning applications. Benefiting from the employment of DNNs in video recognition, many of the previous challenges have been mitigated. Nevertheless, we also encounter new challenges such as long-term dependency modeling in videos and the high computational complexity introduced by DNNs. What is more, video volumes are enormous. Even a short video clip usually contains hundreds of frames. As a result, the computational complexity of a video DNN architecture is far higher than that of images.

In this PhD thesis, we propose practical machine learning algorithms built upon deep learning to challenge the problems encountered in video action recognition. A Convolutional Neural Network (CNN, or ConvNet) is a class of DNNs, which is built by stacking multiple convolutional layers. CNNs have become standard solutions in many computer vision problems. Nevertheless, a standard designed for classifying images is not capable of perceiving temporal changes in videos without employing some special mechanisms or embedding specialized modules. In order to have CNNs learn motion features, Chapter 3 proposes a temporal pooling mechanism, which can map entire video sequences in sets of few 2D images, significantly compressing video representation, while preserving movement information. As a result, a CNN after being configured with our pooling mechanism is capable of learning high-level spatio-temporal features. Intra-class similarities and inter-class variations

represent a challenging problem for pattern recognition tasks. For example, the same person can perform different activities. Or different persons can perform the same activities. Bearing this in mind, Chapter 3 proposes a machine learning algorithm to generate discriminative motion representations for similar videos. Consequently, our model enables accurate action recognition.

Long-term temporal modeling is of importance for action recognition in videos. However, in standard CNNs, long-term modeling is very inefficient, given that convolution is a local operation, whose small kernels would only use the information available within a neighborhood. In contrast, attention [149] and self-attention [131] mechanisms show great advantages for long-term modeling. In Chapter 4, we discuss the usage of attention mechanisms in video recognition. Meanwhile, we propose two novel attention mechanisms to endue CNNs with excellent long-term modeling abilities.

Natural video data contain substantial redundancy in both spatial and temporal dimensions. For example, a static background repeatedly appears in adjacent frames in a video; the positions from a smooth region share the same visual information. Efficient spatio-temporal modeling should take into account the existence of redundant information for optimal results. In Chapter 5, we hypothesize that motion-related information is concentrated in a certain range of spatio-temporal frequencies, while the rest of the frequency bands contain substantial redundancy. After identifying the range of spatio-temporal frequencies considered important for movement representation, and simplifying the video data accordingly, we build a lightweight processing module for efficient video recognition.

1.2 Contributions

(i) The video processing algorithm proposed in the first part of Chapter 3, called Temporal Squeeze Pooling (TSP), is capable of summarizing a long video frame sequence into a few RGB images, which enable existing video models to analyze more video information with fewer computational resources.

(ii) The method proposed in the second part of Chapter 3, called Pixel-Wise Temporal Projection (PWTP), can separate dynamic features from static features in raw video data, resulting higher performance in action recognition tasks.

(iii) The attention mechanisms proposed in Chapter 4, called Region-based Non-local (RNL) operation and Convolution Pyramid Attention (CPA), endue networks with

excellent global context modeling abilities for high discriminate feature learning in videos.

(iv) The Busy-Quiet video disentangling algorithm in Chapter 5 achieves efficient information processing by adopting an intelligent computational resources allocation strategy, which demonstrates superior performance on standard video benchmarks.

1.3 Thesis Outline

The rest of this thesis is organized as follows:

- Chapter 2: We provide a comprehensive literature review for video action recognition methods, going from traditional hand-crafted features to DNN-based methods.
- Chapter 3: We discuss the approaches to spatio-temporal feature representation learning and present two machine learning algorithms for motion feature extraction.
- Chapter 4: We discuss the usage of attention mechanisms in computer vision and present novel attention chains for effectively modeling long-term dependencies.
- Chapter 5: We present a novel methodology for reducing redundant information in the spatial and temporal dimensions and propose a lightweight architecture for fast video recognition.
- Chapter 6: We summarize the works in this thesis and discuss potential future work in video understanding.

Chapter 2

Literature Review

In this chapter, we provide a comprehensive literature review of video action recognition algorithms. Firstly, we introduce the video action recognition datasets which are used for experiments and summarize their statistics. Then, we list the main challenges we encountered. Following this, an in-depth review of video action recognition methods will be provided: going from traditional hand-crafted representation methods to deep learning-based methods. Discussions of these methods including their strengths and weaknesses are also provided.

2.1 Datasets and Challenges

2.1.1 Datasets

We introduce four video action recognition datasets: HMDB51 [73], UCF101 [114], Kinetics [70], Something-Something [48]. The statistics for these four datasets are summarized in Table 2.1.

HMDB51 [73] was made public accessible in 2011, comprising 51 human action categories and containing 6,766 videos. The videos were mainly collected from movies, and a small proportion from public datasets such as the Prelinger archive, YouTube and Google videos. The action categories can be divided into five types: 1) General facial actions; 2) Facial actions with object manipulation; 3) General body movements; 4) Body movements with object interaction; 5) Body movements for human interaction. The dataset is officially split into three parts. The average Top-1 accuracy over the three official splits is commonly reported in many works.

UCF101 [114] was built in 2012, comprising 101 action categories and containing 13,320 videos clips (27 hours) from YouTube. Similar to the HMDB51, the action categories of UCF101 can be grouped into five types: 1) Human-Object Interaction; 2) Body-Motion Only; 3) Human-Human Interaction; 4) Playing Musical Instruments; 5) Sports. It gives a large diversity in terms of actions with the presence of variations in camera motion, dynamic background, illumination conditions, viewpoint, object scale and so on. In order to keep the reported results consistent, three distinct training and testing splits were generated from the dataset. Many works report the classification accuracy averaged over the three splits.

Dataset	# Classes	Clips per Class	Avg. Duration	Total
HMDB51	51	min 102	5s	6,766
UCF101	101	min 101	6s	13,320
Kinetics400	400	min 400	10s	306,245
Something-Something V1	174	avg 620	4.03s	108,499

Table 2.1 Statistics for the video action recognition datasets. "Total" indicates the total number of clips.

Kinetics400 [70] is a large-scale video dataset, introduced in 2017. It contains 240k training videos and 19k validation videos, which are grouped into 400 action classes. Each class has 400-1150 clips. Each clip lasts around 10 seconds, trimmed from a unique YouTube video. The list of action classes covers Person Actions (singular), Person-Person Actions and Person-Object Actions. Most of the actions require more emphasis on the object to distinguish, *e.g*. playing different types of instruments. Some actions require temporal reasoning to distinguish, *e.g*. different type of swimming. Kinetics dataset can be regraded as the successor to HMDB51 and UCF101 which have emerged as the standard benchmarks for this area.

Something-Something V1 [48] was made publicly accessible in 2017. It contains 108,499 clips across 174 human activity classes. The duration of each clip ranges from 2 to 6 sections. The dataset is split into training, validation and testing sets in the ratio of 8:1:1. In Something-Something V1, strong temporal reasoning ability is required to distinguish different action classes, while the object feature is less important. An identical object can appear in multiple videos with different labels, so the videos cannot be distinguished based on just their spatial features.

2.1.2 Challenges

Human activities are of complexity and diversity, which results in the recognition tasks being very challenging. Thanks to the employment of deep neural networks, many challenges inherited from image classification are not as severe as before. Deep network-based methods can significantly reduce the influence caused by viewpoint variation, illumination changes, resolution scale, noise and compression artifacts when using data augmentation skills. The usage of deep learning improved the accuracy of video recognition tasks to an unprecedented level, but it also introduced new challenges. In the following, we list the major challenges of video action recognition tasks in the deep learning era:

- **Inter-class variations and intra-class similarities.** An action can be performed by different people in various scenes. Or an action can be performed at different speeds and recorded with unknown frame rates. What is more, different actions can have similar motion patterns (*e.g*. Brushing teeth *vs*. Shaving Beard). Besides, an individual can perform many different actions in different video clips.
- **Short- and long-term temporal modeling.** Human actions are complicated. Such a complicated human action is made up of multiple atom action. For example, action "Pick up Something" contains atom actions "stretching arms" and "grabbing things " which happen in a temporal order. This requires action recognition models to understand not just short-term motion patterns but also long-term temporal relations.
- **Heavy computational cost.** Deep neural networks acquire knowledge from a large amount of training data. Moreover, video data is collected across both space and time. The computational complexity of video data processing can be a few times that of image data processing.

In this this thesis, each experimental chapter is focused on tackling one of the aforementioned challenges in video action recognition.

2.2 Hand-Crafted Feature Representation Methods for Action Recognition

In this section, we review the hand-crafted feature methods that are closely related to our work.

2.2.1 Appearance-based Temporal Matching

In one of the earliest works in video action recognition [10], Bobick and Davis proposed a classic motion representation and recognition theory that decomposes motion-based recognition into two steps: first describing where the motion happens; second explaining how the motion happens. Meanwhile, they presented an appearance-based method, which constructs a temporal template consisting of a Motion Engine Image (MEI) and a Motion History Image (MHI). A video action can be discriminated by matching the temporal template of the video with the video templates from the training set. Examples of MEI and MHI are shown in Fig. 2.1(b) and 2.1(c), respectively. The MEI is a binary image depicting where motion

(a) (b) (c)

Fig. 2.1 Comparison of RGB frame (a), MEI (b) and MHI (c). Image taken from [11].

occurs in a video sequence. The MHI is a gray-scale image, encoding the temporal order of a video sequence into the 2D space, where older silhouettes have lower intensity while the newest silhouette is overlaid at maximal brightness. Constructing a temporal template by taking the MEI and MHI as two components can represent where and how motion in a video is. The MEI and MHI can be used to recognize simple actions in controlled environments with no camera movement. In order to adapt their method to complex motion analysis, the authors proposed a sequel to the MHI, called the timed Motion History Image (tMHI) [12], which directly encodes the actual time in a floating-point format. Subsequently, they utilize Hu Moment shape descriptors [60] of the silhouette to recognize pose. One shortcoming in the MHI method and its variants is the motion self-occlusion problem, caused by overwriting the temporal information to the same spatial locations, which hinders the modeling of complex motion patterns. In order to circumvent the self-occlusion problem, Ahad *et al*. [1] proposed the directional MHI (DMHI) method, which constructs a temporal template with four directional components by utilizing the information from the four optical flow channels [30] of the video separately. The aforementioned appearance-based temporal matching methods illustrate that motion information in the temporal dimension can be encoded into 2D

images. This idea has become the theoretical basis for many later spatio-temporal modeling algorithms in the deep learning era. For example, built upon the MHI method, the deep learning-based methods, rank pooling [42] and dynamic images [9, 8], are proposed to model movement by aggregating temporal information into a two-dimensional space. Our work described in Chapter 3 of this thesis is also inspired by these works. We will further discuss the ranking pooling and dynamic images methods in Section 2.3.

2.2.2 Space-Time Local Features and Statistical Methods

The aforementioned appearance-based methods belong to the holistic representation family, which retains the spatio-temporal relationships of the voxels. In contrast, spatio-temporal local features are extracted from local 3D patches (key points in the video), omitting the global space-time relationships. Compared with the holistic representations, space-time local features are stable under transition, rotation, scale, viewpoints, background cluttering, illumination changes and other factors produced by uncontrolled environments.

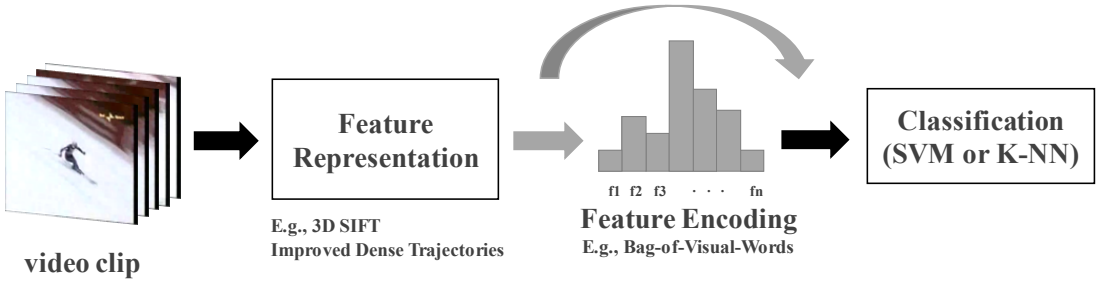


Fig. 2.2 Paradigm of video action recognition using hand-crafted local feature representations.

The general framework of video action recognition using hand-crafted features is shown in Fig. 2.2, in which each video is divided into multiple spatio-temporal patches. local feature algorithms deal with how to represent these local patches as numerical vectors. In order to turn detected local features to fixed-length vectors, local feature-based models usually employ one or multiple local histogram descriptors, operating on each local cuboid. Subsequently, the statistical method Bag-of-Visual-Words converts the feature descriptors to “codewords” by construing a “codebook”. Finally, a video is represented by the histogram of the codewords, which is inputted to a classifier.

The common space-time descriptors include Histogram of 3D gradient (HoG3D) [71] and Histogram of Optical Flow (HOF) [30]. In HoG3D, the 3D gradient is computed at each pixel by differentiating the image function $I(x,y,t)$, and then the 3D gradient is represented by its magnitude and its two orientations. In addition to HoG3D and HOF, there are other descriptors such as Extended SURF [148], Local Trinary Patterns[152], 3D-SIFT [108] or using PCA to generate the descriptor. The Bag-of-Visual-Words method can be replaced by other statistical methods, such as the Vector of Locally Aggregated Descriptors (VLAD) [66], Fisher Vector [105], Kernel Codebook Coding (KCB) [130] and Locality-constrained Linear Coding (LLC) [137]. In the following, we describe the representative methods in the literature of hand-crafted local features.

Space-Time Interest Points (STIP)

Inspired by the sparse spatial interest point operators [44, 51, 107], Laptev [75] devised the Space-Time Interest Points (STIP) method, detecting the "interest points" with high information contents in the spatio-temporal domain. Similar to the behavior recognition paradigm shown in Fig. 2.2, the action recognition systems that use sparse spatio-temporal features are developed in [26, 76]. STIP is based on the detection of spatio-temporal corners

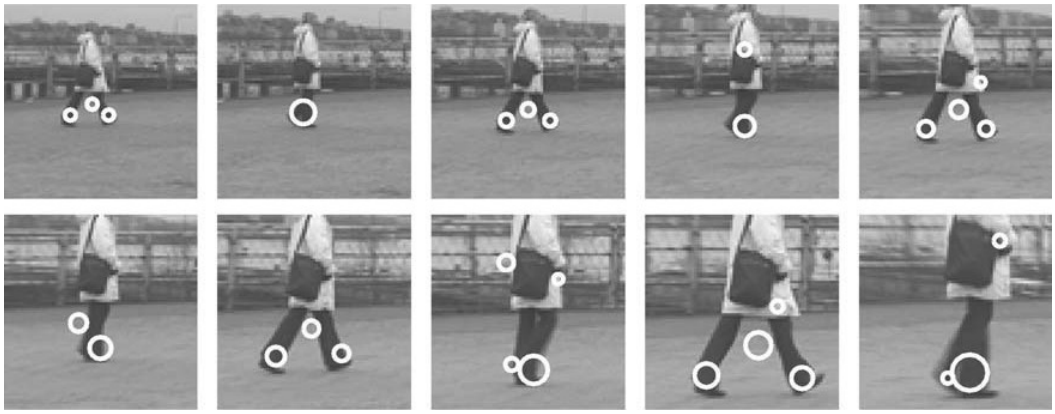


Fig. 2.3 Some examples of Laptev's Spatio-temporal Interest Points. Image from [75].

derived from 2D Harris corner detector [51]. Spatio-temporal corners are located in the regions that exhibit a high variation of image intensity in all three directions(x,y,t) as shown in Fig. 2.3. They are identified from local maxima of a cornerness function computed for all pixels across spatial and temporal scales. After identifying the spatio-temporal interest points, they are represented statistically: a video cuboid would be divided into a set of local cuboid fragments, and each local cuboid will be turned to be a vector with fixed length described by

feature descriptors. The advantage of STIPs is that they are robust to uncontrolled settings, collected in the wild. The disadvantage of STIPs is also obvious: they are detected sparsely and may not capture the whole information from the video. However, Dollar's approach [26] models denser video information by considering all regions in the video. Other interest point detectors include Clouds of Interest Point detector [13] and Hessian-based Spatio-Temporal Interest Point (Hes-STIP) detector [148].

Dense Trajectories (DT)

The spatial dimensions and the temporal dimension in videos have very different characteristics. Intuitively, they should be handled in a different manner. However, the Space-Time Interest Point detectors handle them in the same manner. Encouraged by the success of dense sampling in image classification [35, 96], Wang *et al*. proposed the Dense Trajectories (DT) [133] method, which handles the space domain and time domain in a different way by tracking densely sampled points using optical flow fields. The basic description of dense trajectories is illustrated in Fig. 2.4. Firstly, the initial frame is densely sampled with different spatial scales to get feature points. Secondly, the position of each feature point is tracked to the next frame. The trajectories with sudden large displacement are removed because most likely they are the errors caused by inaccurate optical flow. To make their algorithm robust to camera motion, they utilized Motion Boundary Histogram (MBH) [21] as a descriptor for encoding dense trajectory information. Moreover, the combination of HoG, HOF and MBH provides better performance than that of any of these descriptors. In order to further improve the performance of Dense Trajectories, Wang and Schmid [134] proposed the improved Dense Trajectories (iDT). The most noticeable improvement in iDT is that the trajectories from the background are eliminated by estimating camera motion. Besides, iDT uses Fisher Vector [105] instead of Bag-of-Visual-Words to encode local features. Before 2015, iDT was the dominant solution in a wide range of video understanding applications because of its high robustness, even though some deep learning-based video models were proposed during that time [4, 67, 124].

Some mid-level and high-level representations for video recognition are proposed to overcome the shortcomings of local features, such as Action Bank [104], Dynamic-Poselets [140], Motionlets [139], Motion Atoms and Phrases [138] and Actons [163]. However, these approaches are inferior to Deep Learning features.

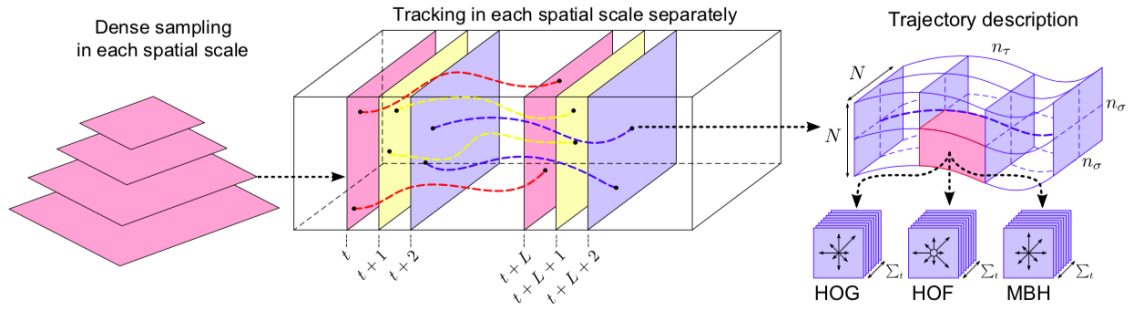


Fig. 2.4 Illustration of the dense trajectory description. Image from [133].

2.3 Deep Learning Era

With the emergence of deep learning [77], the main research direction in video understanding has been shifted from hand-crafted features to deep learning features. Among various deep learning techniques, Convolutional Neural Network (CNN) is the most established deep learning method, which achieves expert-level performance in some computer vision tasks such as handwriting recognition. Fig. 2.5 summarizes the deep learning models for action recognition in a chronological order. From this section to Sec. 2.8, we comprehensively revisit the literature of the deep learning-based action recognition methods from 2014 to the present.

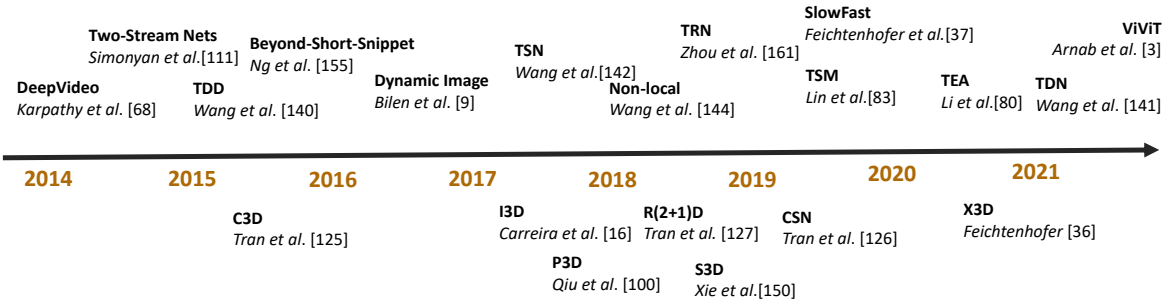


Fig. 2.5 A chronological overview of deep learning-based methods for action recognition in videos.

2.3.1 Convolutional Neural Network (CNN)

In this section, we briefly introduce some important concepts in CNNs. A CNN is a category of artificial neural network, which is commonly used in image classification [72] and other image analysis applications. As shown in Fig. 2.6, a regular CNN consists of an input layer, stacked hidden convolutional layers and fully-connected layers. CNNs have astonishingly

strong feature learning abilities, enabled by their high-nonlinearity and shared-weight nature. A deep CNN is very data-hungry due to its millions of learnable parameters to estimate,

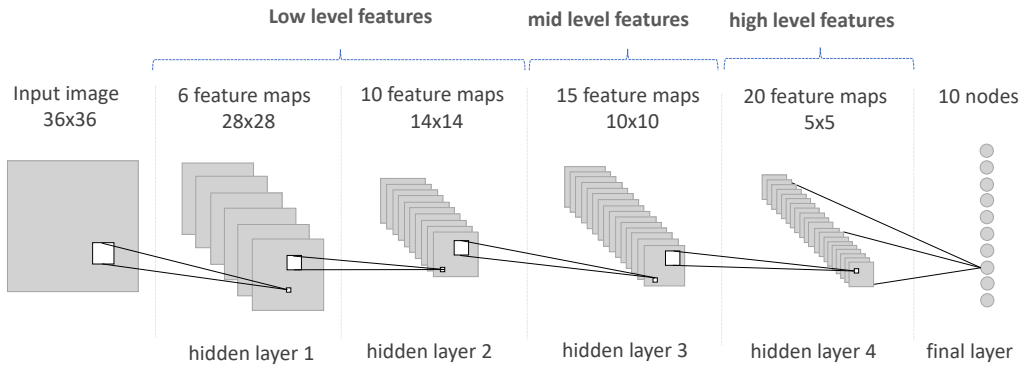


Fig. 2.6 A toy architecture of CNN. As the data in processed further from the input of the CNN, the number of processing filters is increasing, and the size of the output feature maps is getting smaller.

which makes the training of CNN computationally expensive. The parameters of CNNs are commonly optimized with the gradient decent algorithm and backpropagation. However, training a very deep CNN is tricky. Due to the chain rule in backpropagation, shallower layers may receive vanishingly small gradient values, resulting in CNNs' weights updating very slow or even being stalled, which is called "the vanishing gradient problem" [56]. The vanishing gradient problem is a challenge limiting the depth of CNN architectures.

Aside from the vanishing gradient problem, over-parameterization, over-fitting, internal covariate shift [64], and so on are severe challenges in the CNN architecture engineering. To address these issues, several types of CNN architectures were proposed. In Google's Inception architecture [120], the authors unprecedentedly introduced the bottleneck design inception block, concatenating filters of different kernel sizes in one layer to learn multiscale features. The broader width in the Inception layer realistically increases the operative depth of CNN. He *et al*. [54] proposed the seminal ResNet architecture, which is considered a pivotal milestone in deep learning, increasing the maximum depth of CNNs to 152 layers. The main concept in ResNet is referred to as "skip connection", which competently mitigates the training difficulty caused by the vanishing gradient problem. The design patterns of Inception [120] and ResNet [54] have significant impact for latter published works [65, 121, 119, 55, 150, 62, 160]. Furthermore, they are broadly used as backbone networks of diverse models in different tasks, including applications in the natural language processing area.

2.3.2 Single-Frame CNNs and Temporal Fusion

Encouraged by the success of CNN in image classification [72], the pioneering work DeepVideo reported by Karpathy *et al*. [69] attempted to adapt CNNs to video action recognition tasks. By imitating the CNN-based feature learning pattern in image classification, they proposed the single-frame CNN model, in which a single frame from each video is regarded as an individual input sample to the 2D CNN. A single frame contains sufficient static information to describe the scene and object's appearance. However, a single frame carries no temporal information, and it is not sufficient for discriminating the action that happened in the video. Notably, in some large-scale video datasets, the difference of static information between some action classes is subtle, which could hinder the CNN from learning discriminative features to recognize different actions. Distinguishing videos having similar static information relies on modeling motion information in the temporal dimension, which requires that the models are capable of processing multiple frames at a time and learning temporal dependencies between video frames. In order to extend the connectivity of CNNs in the time domain, Karpathy *et al*. [69] further investigate different temporal fusion strategies, including late fusion, early fusion and slow fusion. Even though their model has been integrated with a CNN, which is considered to have a stronger feature learning ability than hand-crafted features in image applications, their performance on UCF101 is still inferior to the hand-crafted iDT features (65.4 *vs*. 87.9%). Nonetheless, the ideas in DeepVideo contributes to later studies.

2.4 Using Optical flow and Two-Stream CNNs

2.4.1 Optical Flow

Directly learning motion features from raw video frames via vanilla 2D CNNs is tricky, as 2D CNNs are originally developed for spatial feature learning. To compensate for this, we can input hand-crafted motion feature representations to a 2D CNN, enabling high-level motion feature learning in the deeper layers of the CNN. Optical flow is a widely used motion representation characteristic, estimating the instantaneous image velocities in the horizontal and vertical directions. In video recognition, the traditional TV-L1 algorithm [157] is commonly adopted for dense optical flow estimation. Some example visualizations of dense optical flow are presented in Fig. 2.7. We can observe that optical flow filters out the appearance information such as static background while clearly indicating the motion patterns of moving objects. In hand-crafted spatio-temporal feature methods, feature descriptors such as HOF [30] and MBH [21] utilize the dense optical flow estimation to generate the targeted

Fig. 2.7 Examples of key frames from videos and their corresponding optical flow extracted by the TV-L1 algorithm [157]. We use the same color rendering scheme as in [6] for visualizing the optical flow.

features.

2.4.2 Two-Stream Networks

Drawing upon the two-pathways hypothesis [47] of the human visual system with one stream representing objects and the other stream representing motion, Simonyan *et al.* [112] proposed the two-stream architecture. As shown in Fig. 2.8, the two-stream architecture consists of a spatial stream CNN and a temporal stream CNN, which are of the same design except for their input layers. The spacial stream processes individual frames while the temporal stream takes a stack of optical flow images as input. Precisely, the optical flow images in the temporal stream are calculated by the TV-L1 algorithm [157], the values of which are re-scaled to $[0, 255]$, and then are stored in an image compressed by JPEG. At the end of the two-stream architecture, the prediction scores of the spatial and temporal streams are averaged to produce the final prediction. In comparison with a single spatial CNN, the two-stream fusion improves the accuracy on UCF101 from 73.0% to 88.0%, which is even higher than that of iDT (87.9%). Their work narrowed the gap between the state-of-the-art hand-crafted features and deep learning-based methods. In addition, the two-stream CNNs reveal the importance of motion features for action recognition, even though learning motion features from raw RGB frames is still challenging for 2D CNNs. The idea of two-steam CNNs has become a major research direction in video understanding. Many later published works followed the two-stream idea to develop their models.

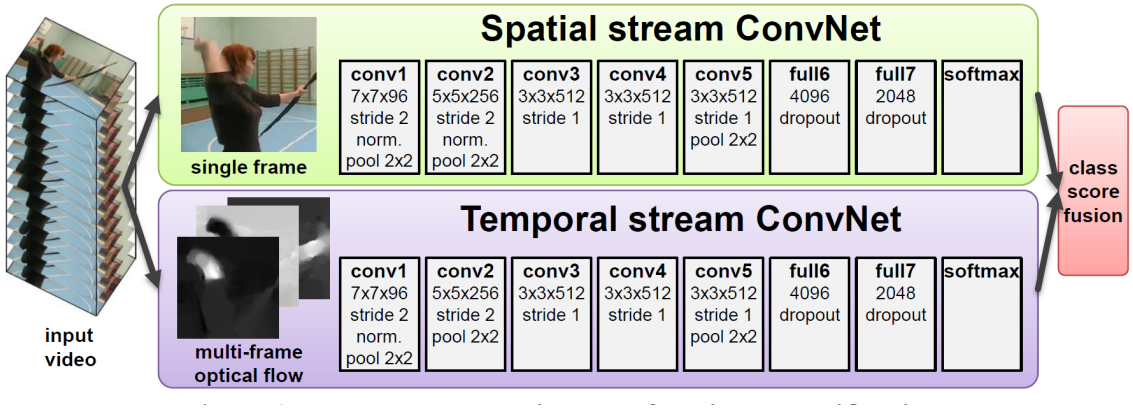


Fig. 2.8 Framework of the two stream architecture. Image from [112].

To further increase the performance of two-stream CNNs, we can adopt a deeper backbone network, such ResNet [54] or 3D CNN [67]. On the other hand, the quality of optical flow estimation, which is used as input to the temporal stream, has an influence on performance to some extent. Improving the optical flow estimation with a better algorithm can be a way to increase the accuracy for action recognition in videos. One issue that limits the efficiency of two-stream CNNs [112] is that the optical flow images are not produced on the fly, which makes the deployment of two-stream CNNs difficult in real-world applications requiring low latency. In Sec. 2.6.3, we will further discuss the methods for efficient optical flow estimation in video recognition. A deep CNN require a large amount of training data to avoid over-fitting. Labeling video data is a very laborious operation. Since each stream in the two stream architectures is a 2D CNN, its parameters can be initialized with the model trained on some large image dataset such as ImageNet [22]. Although the distribution of optical flow is different from that of RGB images, the weight of each channel in the first layer of the temporal stream can be initialized by the average of the weights across the RGB channels, according to the study from [143].

2.4.3 Strategies for Two-Stream Fusion

In the two-stream networks [112] and the models that are composed of two or more streams [143, 16, 9, 33, 164, 84, 158], each stream is trained independently. With the aim of utilizing the features from these independent streams when performing prediction, the Softmax outputs of different streams will be fused by a certain operator. The type of fusion strategy that merges Softmax outputs at the end of all streams is called late fusion. In [40], late fusion using max, convolution and concatenation operators are further explored. Feichtenhofer *et al.* [40] point out that training each stream individually and the late fusion

may not fully exploit the correlations between motion and appearance information in the early stage. To rectify this, they proposed a unified convolutional two-stream network where the two streams are trained in a joint manner, and the spatial and temporal cues are fused at several different abstraction levels. They concluded that the best place for the spatio-temporal fusion is right after the last convolutional layer. Inspired by ResNet [54], Feichtenhofer *et al*. [38, 39] further proposed two sequels to the convolutional two-stream network by introducing residual connections between the two-streams and a multiplicative gating function to their models for a better appearance and motion information fusion. Meanwhile, Wang *et al*. [147] proposed a spatio-temporal pyramid network supported by long-term temporal modeling and a visual attention mechanism, hierarchically fusing the spatial and temporal information.

2.4.4 Sparse Frame Sampling and Temporal Relational Reasoning

Considering that a video is usually composed of hundreds of frames, it is not computationally feasible to process all video frames at once with limited computational resources. Hence, the approach to sample video frames becomes vital to performance in terms of both efficiency and effectiveness. In [69], the single-frame CNN each time processes only one RGB frame sampled from a video randomly, resulting in poor action discrimination abilities due to the lack of temporal information. Although the two-stream CNNs [112] have enabled the short motion feature modeling by using stacked optional flow images as the additional input modality, the long-range temporal structure is still not captured appropriately. In order to capture the global temporal evolution in videos with no need to input all video frames, Wang *et al*. [143] proposed Temporal Segment Networks (TSNs). Specifically, TSN adopts a sparse sampling strategy in the temporal dimension, evenly dividing an entire video into multiple segments. Subsequently, TSN randomly picks one frame for each segment and then processes the selected frames with the CNNs. In the end, all frame-level predictions from all CNNs are fused to generate the video-level predictions. Furthermore, TSN adopts a multi-stream design, building additional streams by taking stacked optical flow and RGB difference as the inputs. The framework of TSN is straightforward but efficient for long-temporal structure modeling, credited to the sparse frame sampling strategy which provides a global view of sparse temporal information in videos.

Encouraged by the simplicity and efficiency of sparse frame sampling in TSN [143], many segment-based CNN methods [82, 46, 74, 25] are proposed. However, TSN has not taken the temporal order of video frames into consideration, which means input frames with correct temporal order and that in shuffled order have an identical output in TSN. The temporal

relations in videos has less influence on performance on "scene-dominant datasets" such as Kinetics [70] but is vital to "motion-dominant datasets" such as Something-Something [48]. In order to give CNNs a capacity to discover temporal relations in videos, Zhou *et al*. [162] proposed the Temporal Relation Network (TRN), which can learn and reason about temporal dependencies between video frame at multiple time scales. Inspired by the RGB difference input proposed in TSN [143], TDN [142] devises an efficient temporal module by leveraging a temporal difference operator to fully capture temporal information over the entire video. Recent temporal segment-based methods include TSM [84] and TEA [81], more details of which will be discussed in Sec. 2.7.3.

2.5 2D CNN + Recurrent Neural Network (RNN)

2D CNNs are satisfactory spatial information handlers with excellent semantic-level feature learning ability but not gifted to extract the dynamics within frames. Nevertheless, Recurrent neural networks (RNNs) are well-suited to process time series data with no fixed length, which can compensate for the shortcomings of 2D CNNs. An idea for video action recognition is to take advantage of 2D CNNs and RNNs to build a unified recognition framework, capturing the appearance and motion features as a whole.

LSTM is a class of RNNs, which is particularly developed to deal with the vanishing gradient problem [56] which can be frequently encountered when training traditional RNNs. An LSTM unit generally consists of an input gate, a memory cell, a forget gate and an output gate. Beyond-Short-Snippets [156] is one of the first deep learning models that adapt LSTMs to video understanding problems. The CNN-LSTM model is capable of learning the global temporal evolution of videos. As shown in Fig. 2.9, the CNN-LSTM model first uses a 2D CNN (red rectangle) to individually process each consecutive frame. Subsequently, the CNN outputs are processed forward through time via stacked LSTMs (purple cuboid). After the top LSTM layer, a Softmax layer (orange rectangle) outputs the scores at each timestamp. Finally, a late fusion function is adopted to aggregate the frame-level Softmax scores into a video-level prediction. Another notable point of the CNN-LSTM model is that the input videos are not restricted to fixed lengths, given that RNNs can model variable-length time series data. Concurrently, Donahue *et al*. [27] proposed Long-Term Recurrent Convolutional Networks (LRCNs) with a very similar idea as in [156]. Drawing upon the CNN-LSTM model [156], later works proposed diverse variants including two-stream LSTM [45], TS-LSTM [91], bi-directional LSTM [129], VideoLSTM [83] and Lattice-LSTM [116]. To model long-term dependencies for action recognition, a biologically-inspired deep network

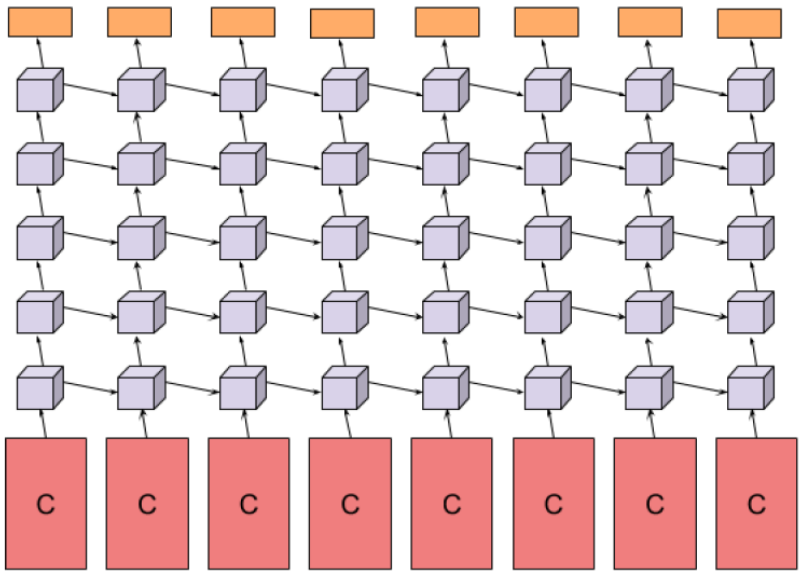


Fig. 2.9 The CNN -LSTM [156] architecture for action recognition in videos. Image from [156].

is proposed in [111], in which all GRUs are cyclically connected mimicking the biological neural system that commonly contains both feedforward and feedback connections. In [16], Carreira *et al*. provide a thorough evaluation for the two-stream CNNs and CNN-LSTM methods, both of which adopts Inception-V1 [120] as the backbone networks for a fair comparison. They empirically demonstrate that the single stream CNN-LSTM has significantly lower accuracy than the two-stream CNNs. Nevertheless, the CNN-LSTM model can still take optical flow as an additional input modality and perform two-stream fusion with its spatial part as done in [156]. The two-stream CNN-LSTM model has better performance over any of the CNN-LSTM and two-stream CNNs models, according to the empirical results in [156, 91, 27].

2.6 Motion Representation Learning

2.6.1 Trajectories Meet Deep Learning

2D CNNs can learn high-level semantic features in the spatial domain, but are poor at capturing the temporal information in videos, unless we consider additional features such as the optical flow. Nevertheless, the temporal relations may still not be properly modeled, as we have discussed in Sec. 2.4.4. Traditional hand-crafted features such as improved Dense Trajectories (iDT) [134] present stable performance when modeling temporal evolution in

videos. In order to capture long-term motion in video frame sequences, Wang *et al*. [141] proposed a new type of video representation, called trajectory-pooled deep-convolutional descriptor (TDD), which shares the merits of both deep learning and iDT. Briefly, they employ the two-stream CNNs [112] to generate high-level convolutional feature maps and then conduct the strategies of trajectory-constrained sampling and pooling to aggregate the convolutional feature maps over a sequence of frames into efficient descriptors. To suppress the over-activation of the neurons in the deep learning architecture, they apply spatio-temporal normalization and channel normalization to the convolutional feature maps. As to the feature encoding, the TDDs of the entire video are encoded by Fisher vector [105] to form a global super vector. In the end, the global super vector is taken as the input to a linear SVM to carry out the classification. The TDD method was the state-of-the-art in 2015, demonstrating higher performance than iDT [134] and the two-stream CNNs [112] on UCF101 [114] and HMDB51 [73]. One shortcoming in the TDD method is that the whole model is not an end-to-end framework, in which the CNNs are considered to be a fixed feature extractor pretrained on a large-scale image dataset. Lately, Zhao *et al*. [161] proposed trajectory convolution which is capable of integrating features along the temporal dimension. By incorporating spatial 2D convolution, trajectory convolution can capture features in both spatial and temporal dimensions, which can be used to replace the 3D convolution operations in 3D CNNs [67].

2.6.2 Pooling Functions for Motion Pattern Modeling

Inspired by the traditional holistic video representation Motion History Image (MHI) [10] and rank pooling [42, 43], Bilen *et al*. [9] devised a novel holistic video representation, termed dynamic images. Rank pooing [42] is originally designed to learn the video-wide temporal evolution and return a ranking function for each video. As the ranking function is capable of arranging the video frames in chronological order, its parameters can be used to construct a video descriptor. Bilen *et al*. [9] adapt rank pooling to deep learning and operate on multiple consecutive raw RGB frames, resulting in a dynamic image. A dynamic image is an RGB image with 3 color channels that summarize the appearance and motion information from consecutive video frames. Dynamic images can be used as inputs to a 2D CNN. Consequently, the whole framework is capable of modeling the spatio-temporal information in videos. Following this, Wang *et al*. [136] proposed SVM Pooling (SVMP) which uses the decision boundaries of 2 class SVMs as a new video representation. Similar to dynamic images [9], the video representation generated by the SVMP is also an RGB image. A visualization example that compares dynamic images and SVMP is shown in Fig. 2.10. It can be observed that SVMP captures more motion details about the action that

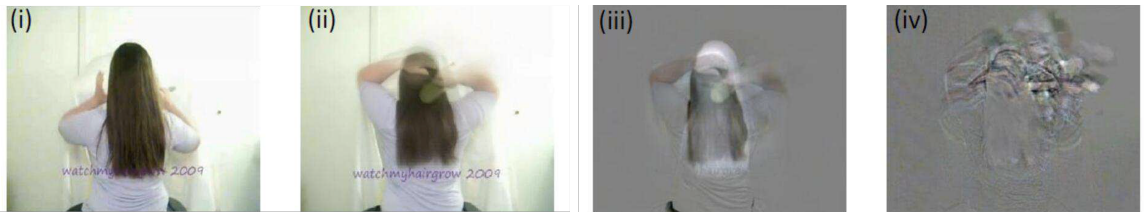


Fig. 2.10 Visualization of the results of SVM pooling and approximate rank pooling when applied to a sequence of video frames. (i) a sample frame, (ii) average pooling all frames, (iii) dynamic image by approximate rank pooling and (iv) SVM pooling. Image from [136].

the dynamic image. Both rank pooling and SVM Pooling are essentially dimensionality reduction techniques, projecting video data from the spatio-temporal domain to a reduced spatial domain. Meanwhile, the usages of ranking pooling and SVM pooling bring us to the conclusion that video data across space and time contain significant redundancy which does not improve the performance but increases the computational complexity of models. One reason is that the background information does not usually change from frame to frame.

2.6.3 End-to-End Flow Estimation and Flow-like Methods

One major reason that limits the efficiency of two-stream CNNs [112] is that the optical flow images in the temporal stream are not produced on the fly. The dense optical flow has to be pre-computed and stored locally. Additionally, the computation of dense optical flow is relatively time-consuming. Many researchers have been thinking of how to address these problems. In order to estimate optical flow inside a unified framework for video action recognition, the published works [24, 95] attempted to integrate an optical flow estimation network into the recognition model, training in an end-to-end manner. Although the inference process of these methods no longer involves pre-computed optical flow, these methods still require ground-truth optical flow to train the flow estimation networks inside their models. The work in [95] also designs a stacked model, in which a 2D CNN is at the top of the pretrained optical flow estimation network FlowNet [63]. Benefiting from the pretrained parameters in FlowNet, the stacked model does not require pre-computed optical flow anymore. Nevertheless, the extra computation required by FlowNet is not trivial, resulting in twice more computational cost for inference. The stacked structure is also adopted in hidden two-stream networks [164]. Diversely, hidden two-stream networks replace the heavy optical flow estimation network with their MotionNet, which is relatively lightweight and trained in an unsupervised manner. As a result, hidden two-stream networks require no additional pre-computed optical flow in training and inference. Mocking the motion pattern in optical flow, Fan *et al*. [33] proposed TVNet, which approximates the traditional optical flow

estimation algorithm TV-L1 [157] in a differentiable way. TVNet can be regarded as a fully-differentiable layer embedded into a video network to form an end-to-end framework, thus avoiding the pre-computation and the requirement for local optical flow storage. Following this direction, the published works [118, 78, 100, 158] proposed their own flow-like motion features, which are end-to-end trainable. MARS [20] proposed to combine appearance and motion information into a single stream. They design a knowledge distillation method to transfer knowledge from the optical flow stream to a network with only RGB input. Similar work was also done in [115], leveraging a teacher-student model.

2.7 3D Convolution-based Methods

2.7.1 3D Convolutional Neural Networks (3D CNNs)

A video can be conceptually regarded as a 3D tensor with two spatial dimensions and a temporal dimension. The straightforward way to process 3D tensors is to use 3D convolution filters, given that 3D convolution can simultaneously process temporal and spatial information. Motivated by this, the pioneering work by Ji *et al*. [67] employs 3D convolution to build an action recognition model, termed 3D Convolutional Neural Network (3D CNN). The 3D CNN is composed of a hardwired layer, three 3D convolutional layers and a fully-connected layer. In particular, the hardwired layer contains a set of predefined filters, generating the gray, gradient-x, gradient-y optflow-x and optflow-y, which can accelerate the training in the early stage and provide better performance when compared with random initialization. On the basis of the 3D CNN model, Tran *et al*. [126] proposed a deeper network with 3D convolutions, named C3D. The C3D architecture can be regarded as a 3D extension of the VGG16 [113]. Although 3D CNNs provided a promising direction for spatio-temporal modeling, their performance was not competitive with that of the state-of-the-art. Because of the additional kernel dimension, 3D CNNs contain many more parameters than two-stream CNNs and CNN-LSTM models, resulting in extended training periods. Additionally, 3D CNNs are prone to over-fit on small-scale datasets such as UCF101 [114] and HMDB51 [73], having lower performance than two-stream CNNs.

Before 2017, the two-stream CNNs method was the dominant solution in action recognition. In 2017, Carreira *et al*. [16] proposed the Inflated 3D ConvNet (I3D) based on Inception-V1 [120] and 3D operations. The authors made a comprehensive comparison over a set of video action recognition models, including CNN-LSTM, two-stream 2D CNNs, 3D CNNs and two-stream 3D CNNs. The I3D model achieved close performance to the two-stream

2D CNNs, and their two-stream 3D CNNs outperformed all the other models on several standard video benchmarks. Compared with 2D CNNs, 3D CNNs are more data-hungry but they also have higher feature learning capacity. In order to fully exploit the feature learning ability of 3D CNN, the authors of I3D experimented on the large-scale dataset Kinetics400 [70] whose training period can last a few weeks. It is no doubt that this made the research cycle time longer and required more GPU machines, making the computational requirements only affordable for big companies and organizations. Nevertheless, Hara [50] claims that a large volume of annotated data is indispensable for fully optimizing the massive parameter numbers of 3D CNNs. Meanwhile, they proposed 3D ResNets by simply replacing the 2D kernels in ResNet [54] with 3D operations. They demonstrated the performance of 3D ResNets on the Kinetics dataset can be effectively improved by stacking more 3D convolutional layers, until reaching the depth of 152 layers.

2.7.2 3D Factorization

Since the great improvement made by I3D [16] in action recognition, many researchers focus on investigating new 3D CNN-based architectures. The aforementioned 3D CNNs [126, 16] are too computationally expensive. In order to simplify the 3D CNNs, Qiu *et al*. [101] devised Pseudo-3D (P3D) Networks, where a $k \times k \times k$ 3D convolution is factorized into a $1 \times k \times k$ spatial convolution and a $k \times 1 \times 1$ temporal convolution, which significantly reduces the computational complexity. Similarly, Tran *et al*. [128] proposed another 3D factorization method, termed R(2+1)D. The main difference between P3D and R(2+1)D is that an R(2+1)D network is a homogeneous architecture built with a single type of (2+1)-decomposition while a P3D network includes three different types of residual blocks. Before the proposals of P3D and R(2+1)D, Sun *et al*. [117] in 2015 already had the idea about spatio-temporal factorization for action recognition. However, due to the insufficient training data in the annotated video datasets that they experimented on, the capacity of their model called F_{ST}CN was not fully explored. Xie *et al*. [151] apply (2+1)D-factorization to I3D [16] and seek a trade-off between speed and accuracy by only replacing some 2D convolutions in Inception Networks [120] with 3D convolutions. Their top-heavy model, which has 2D operations at the bottom and 3D operations at the top, generates higher accuracy than their bottom-heavy model where the location of 3D and 2D operations are inverted. This suggests that modeling temporal patterns in high-level features with rich semantics is critical for improving performance in video understanding.

2.7.3 Towards Efficient and Effective Spatio-Temporal Modeling

The innovative 3D factorization methods described above were proposed for efficient video understanding. In order to further increase the efficiency of 3D Networks, some published works with creative ideas different from 3D factorization were recently proposed. In image classification, efficient 2D networks such as Xception [19], MobileNet [58] and ShuffleNet [160] achieve excellent trade-off between accuracy and speed by utilizing 2D depthwise separable convolution. Encouraged by this, Tran *et al*. [127] introduced the concept of 3D channel-separated convolution to video action recognition. The 3D channel-separated convolution is an instantiation of group convolution, which provides a good practice to separate the channel and spatio-temporal interactions. Their 3D Channel-Separated Convolutional Network (CSN) [127] achieved higher accuracy than conventional 3D CNNs while still being 2-3 times faster. Inspired by the fact that the motion being processed can evolve at a slow or fast speed, Feichtenhofer *et al*. [37] proposed an efficient network, termed SlowFast, which is composed of a slow pathway and a fast pathway. The Slow pathway operates at a specific low frame rate in order to capture spatial semantics while the Fast pathway operates at a specific high frame rate to capture motion at a fine temporal resolution. The two pathways are linked with multiple lateral connections for fusing the learned features between the two pathway. In order to reduce the computational cost, the Fast pathway is implemented with a network of low channel capacity and a lower spatial resolution. The SlowFast architecture is considered to be closely related to the two-stream CNNs [112], but their processing approach towards spatial semantic and temporal motion modeling is clearly distinct from each other. What is more, SlowFast does not rely on optical flow to model temporal information. Similarly, bLVNet [34] and Octave Convolution [18] were proposed to factorize the feature maps outputted by spatio-temporal convolutions in order to reduce the computational cost. For efficiently modeling spatio-temporal information Lin *et al*. [84] proposed a generic module, called temporal shift module (TSM), which is conceptually a zero FLOP operation, introducing no additional parameters. The TSM enables motion pattern modeling by shifting a portion of channels along the temporal dimension, such that it facilitates information exchange among neighboring video frames. The best merit of TSM is its plug-and-play property which can be easily embedded into off-the-shelf 2D CNN architectures for temporal modeling. In the Temporal Excitation and Aggregation (TEA) module [81], which is for modeling long-term temporal dependencies, the temporal convolution is implemented in a similar way to TSM.

Neural architecture search (NAS) is used to design networks that are on par or outperform hand-designed architectures in image classification [166, 123, 122]. Particularly, Efficient-

Net [122] leverages NAS to design an auto network and scales it up to obtain a family of instantiations to meet different required complexities. Based on this, Feichtenhofer [36] proposed a model scaling method for discovering efficient video network architectures. The resulting architecture called X3D (Expand 3D) is obtained by progressively expending a tiny 2D network along multiple network axes, in space, time, width and depth, until the targeted complexity is reached. X3D achieves similar accuracy as previous work while requiring 4-5 times fewer FLOPs.

2.8 Transformers-based Methods

2.8.1 Vision Transformers (ViTs)

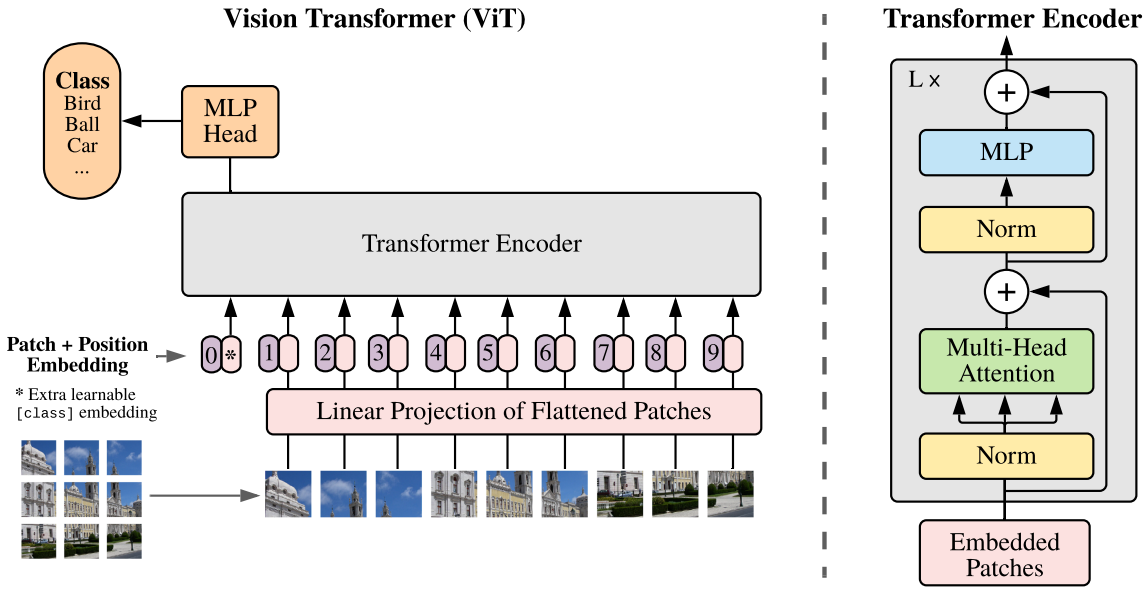


Fig. 2.11 The framework of Vision Transformer (ViT). Image from [28].

A transformer [131] is a sequence-to-sequence (Seq2Seq) architecture that adopts the self-attention mechanism, being the de-facto standard for natural language processing (NLP). The dominance of Transformers in NLP is just like that of CNNs in computer vision. However, we are witnessing a shift in backbone architectures for computer vision from CNNs to Transformers. This trend starts with the emergence of the Vision Transformer (ViT) [28], which globally models spatial dependencies on 16×16 spatial patches with the Transformer in NLP [131]. As shown in Fig. 2.11, ViT is built by stacking multiple-head attention modules and MLPs. One of the key concepts in Transformers is self-attention whose advantage is to allow the network to recalibrate the significance of each token. In

other words, the self-attention mechanism provides networks a global view of the entire input as its output establishes the relationships between different tokens. ViT outperforms previous CNN-based architectures [54, 119, 120] in image classification. Following this, many scholars have joined this vision modeling revolution, proposing diverse ViT-based models [125, 154, 49, 144, 86] showing superior performance to CNNs. The tremendous success of ViT in image classification has triggered the investigation of Transformer-based architectures for video action recognition [3, 7, 159, 32, 94]. Taking the video Transformer ViViT [3] as an illustrative example, the authors radically expand the ViT [28] from space to spacetime: They extract spatio-temporal tokens from the input video, which are then encoded by a series of transformer layers. The weight parameters of ViViT are initialized by the ViT model pretrained on a large-scale image dataset to enable faster convergence in its training.

2.8.2 ViTs vs. CNNs

According to the empirical evidence in [28], CNNs still provide higher performance than ViTs when the pre-training datasets are not large enough. As the dataset size increases, the accuracy of ViTs gradually catches up and surpasses CNNs. This phenomenon suggests that Transformers are more data-hungry than CNNs. Due to a great number of stacked self-attention mechanisms in Transformers, the computational complexity of ViT is massive, leading to very long training times. The unsatisfactory performance of CNNs may be due to the old-fashioned design and outdated training regimes. In the latest work [88], the authors reexamine the design spaces and test the limits of what a pure CNN can achieve. Their ConvNeXt, which is pure convolutional architecture, demonstrates higher performance than previous Transformer-based architectures in image classification. As vision transformers were proposed very recently, the research methods presented in this thesis and their implementation do not consider these ViT-based backbone architectures.

2.9 Conclusion

In this chapter, we have provided the statistics of the datasets on which we experiment from Chapter 3 to Chapter 5. Then we have listed the main challenges for video action recognition. This PhD thesis will take practical measures to deal with these challenges in the following experimental chapters. In this chapter, we have presented the development of action recognition in videos, going from using traditional hand-crafted feature representation methods to deep learning-based approaches. Regarding deep learning-based approaches, we have roughly divided them into six classes, including single-frame networks, two-stream

networks, 2D CNN + RNN, motion representation learning, 3D convolution-based methods, transformer-based methods. Most methodologies from different classes overlap each other, having many elements in common. For all these methods, we have introduced their key concepts and critically analyzed their advantages and disadvantages, including discussions of how their ideas have inspired the later works. Although the experiments of this thesis have not involved Vision Transformers, we have also discussed the literature of transformer-based methods.

Chapter 3

Spatio-temporal Feature Representation Learning

3.1 Introduction

Extracting feature representations characteristic to the movement in the scene is essential for video tasks but still challenging. From the literature review in Chapter 2, the performance of traditional hand-crafted feature representations is unsatisfactory, to some extent, in comparison with the deep features. Nevertheless, combining hand-crafted feature representation methods with the deep learning technique [112] shows us a promising way for efficient spatio-temporal feature learning. Fig. 3.1 illustrates the general process for action recognition

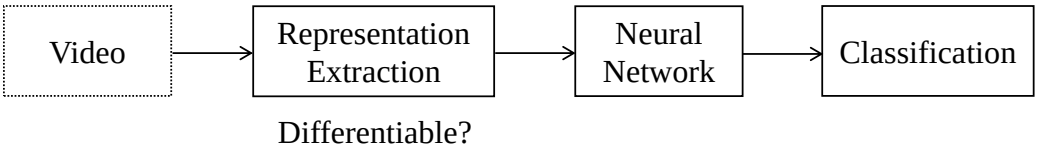


Fig. 3.1 The general process for action recognition.

in published works [112, 143, 8, 33]. In the first step, the model utilizes a spatio-temporal representation extraction method for capturing the visual representations of videos. After that, the extracted video representations are taken as input by a neural network for learning abstract feature maps that encode the motion information. The preliminary representation extraction in the first step aims to increase the efficiency of spatio-temporal feature learning of the neural network in the later step. In some published works [8, 100], their representation

extraction process is embedded in some layers of different depths in the neural network, but the purpose is the same.

As shown in Fig. 3.1, if the representation extraction module is differentiable, then the action recognition process is an end-to-end training pipeline, which means the model generates motion representations on the fly, bringing great flexibility for the classification framework. Furthermore, if the representation extraction method contains trainable parameters, the parameters of this representation learning method would be optimized with the criteria of action recognition, as such, the generated video feature representations would be generalized to many different video scenes, which as a consequence benefit the performance of action recognition and downstream tasks.

Following the general process of action recognition in Fig. 3.1, this chapter focuses on designing efficient and effective video representation extraction methods. This chapter can be divided into two main parts:

1) In the first part, starting from Section 3.2, we describe a novel video representation, termed Squeezed Image, which summarizes the dynamics and appearance information of a long video sequence. Meanwhile, we introduce a new temporal pooling approach, called the Temporal Squeeze pooling (TSP), which can be embedded into off-the-shelf CNNs, allowing the training of the whole model in an end-to-end manner.

2) In the second part, starting from Section 3.5, we introduce the concept of Dynamic Appearance. By upgrading the temporal squeeze pooling to a pixel-wise operation, named Pixel-Wise Temporal Projection (PWTP), we successfully separate the dynamic appearance of a video from its static appearance. In action recognition tasks, the neural networks that take dynamic appearances as input could avoid the distraction caused by irrelevant static information during the training and thus, produce superior performance.

In the end, we draw a conclusion that summarizes the contribution of this chapter.

3.2 Squeezed Image: End-to-End Representation Learning

Inspired by the classic appearance-based method [10] which characterizes motion by constructing 2D silhouettes for videos, we attempt to encode the dynamics of a long video

sequence into a few 2D images. Our encoding scheme enables invisible motion clues in the temporal dimension to visually present in 2D images, such that 2D CNNs can directly learn high-level motion features from these extracted video representations without requiring additional 3D operations. The work closely related to ours is [8], in which the authors introduce the concept of dynamic image. A dynamic image is an RGB image that summarizes the appearance as well as the dynamic information of videos. A dynamic image is constructed as the parameters of a rank or approximate rank pooling machine learned to sort the temporal order of video frames. The dynamic image accelerates the classification process of neural networks by reducing the analysis of a video to the analysis of a single image. However, we reveal that the information storage capacity of a single image is very limited and, as such, a dynamic image is not capable of representing the dynamics of a very long video sequence. To fix this problem of dynamic images, Bilen *et al*. [8] divide the long video sequence into several sub-sequences of shorter lengths. However, this approach is still not able to effectively improve their models' performance.

Here, we introduce a new video representation, named Squeezed Image. Meanwhile, we propose a novel pooling method, named Temporal Squeeze Pooling (TSP), to generate the squeezed images. The proposed TSP aggregates the temporal video information into a reduced spatial dimension by means of an optimization approach that preserves the video information characteristics. In this study, TSP is optimized for the action recognition task. TSP can compensate for the shortcomings of the rank pooling machines [42, 43, 9, 8], by controlling the pooling size. We demonstrate that the proposed TSP mechanism can summarize the visual representation of up to 64 video frames with highly-complex temporal changes while the rank pooling machine is limited to processing only 10 frames without performance degradation. When processing 20 video frames, the ranking pooling machine resulted in a significant performance drop, according to the experiment in [8]. By embedding the TSP module as a layer into the off-the-shelf CNNs, we design a new action recognition model named the Temporal Squeeze Network (TeSNet). The proposed TesNet architecture significantly reduces the overhead in processing video data while improving the accuracy in action recognition.

3.2.1 Temporal Squeeze Pooling (TSP)

In this section, we first describe Temporal Squeeze Pooling (TSP), which can aggregate the long-term temporal information of a sequence of video frames into a small set of squeezed images. Then, we show how we embed the TSP block as a layer into a CNN, forming an end-to-end training pipeline.

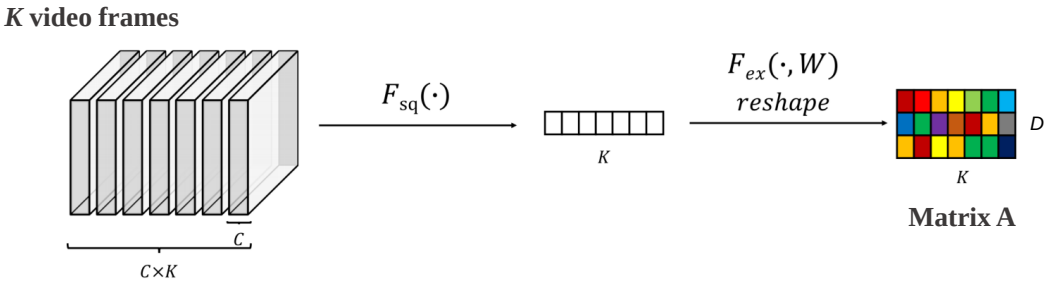


Fig. 3.2 Diagram of squeeze and excitation operations in the temporal squeeze pooling. For each video frame, the squeeze operation $F_{sq}(\cdot)$ aggregates the information of all positions and represents it with a single value. Then the excitation operation $F_{ex}(\cdot, \mathbf{W})$ consumes the vector of k values to generate $\mathbf{A}$, which would be used to construct the squeezed images later.

The proposed approach relies on the observation that consecutive video frames usually contain repeating information, especially either the background for a still camera, or the foreground, when the camera follows a target. A temporal squeeze pooling module aims to compress the dynamic information of a video clip with K frames into D frames ($D << K$), such that essential information is preserved. Consequently, repeating information is filtered out while preserving the essential, usually specific movement patterns. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K]$ denote K video frames where $\mathbf{x}_i \in \mathbb{R}^{H \times W \times C}$, $i = 1, \ldots, K$ and H, W, C represent the height, width and the number of channels (color), respectively. The TSP aims to find out the optimal hyperplane determined by the column space of matrix $\mathbf{A} \in \mathbb{R}^{K \times D}$, and then map every pixel of $\mathbf{X}$ from the vector space of $\mathbb{R}^K$ onto a much smaller information defining space $\mathbb{R}^D$. The aim is to preserve the relevant dynamic information across the temporal direction into the compressed space.

In the following, the squeeze and excitation operations proposed in [59] are adopted for the TSP. The frame sequence $\mathbf{X}$ is initially processed by the squeeze operation, producing a frame descriptor. The squeeze operation is implemented by using global average pooling along the spatial dimensions H, W and C. Then, the squeeze operation is followed by the excitation operation, which is made up of two fully connected (FC) layers. The output of the excitation operation is reshaped into $\mathbf{A} \in \mathbb{R}^{K \times D}$, which defines a hyperplane in $\mathbb{R}^K$. In the squeeze operation, the k-th element of frame-wise descriptor $\mathbf{z} \in \mathbb{R}^K$ is calculated by:

$$z_k = F_{sq}(\mathbf{x}_k) = \frac{1}{HWC} \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{l=1}^{C} x_k(i, j, l). \tag{3.1}$$

In the excitation operation, the input-specific hyperplane is calculated by:

$$F_{ex}(\mathbf{z}, \mathbf{W}) = \delta_2(\mathbf{W}_2\delta_1(\mathbf{W}_1\mathbf{z})), \tag{3.2}$$

where δ_1 and δ_2 refer to the activation functions and $\mathbf{W}_1 \in \mathbb{R}^{K\times K}$, $\mathbf{W}_2 \in \mathbb{R}^{KD\times K}$ refer to the weights of the FC layers. Then, the output of Eq. (3.2) is reshaped into a matrix $\mathbf{A}' \in \mathbb{R}^{K\times D}$. The input-specific $\mathbf{A}$ for the projection is given by

$$\mathbf{A} = \Phi(\mathbf{A}'), \tag{3.3}$$

where Φ is a function that guarantees the columns of $\mathbf{A}$ are linearly independent. We flatten $\mathbf{X}$ along its H, W and C dimensions into a vector $\overline{\mathbf{X}} = [\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, ..., \bar{\mathbf{x}}_{HWC}]$ where $\bar{\mathbf{x}}_i \in \mathbb{R}^K$, and then project it onto the column space of $\mathbf{A}$, resulting in a vector $\widehat{\mathbf{X}}$. The i-th element of projection $\widehat{\mathbf{x}}$ is calculated by:

$$\begin{aligned} \mathbf{y}_i &= (\mathbf{A}^\intercal\mathbf{A})^{-1}\mathbf{A}^\intercal\bar{\mathbf{x}}_i, \\ \widehat{\mathbf{x}}_i &= \mathbf{A}\mathbf{y}_i, \quad i = 1, \ldots, HWC \end{aligned} \tag{3.4}$$

where $\mathbf{y}_i \in \mathbb{R}^D$ represent the mapping coefficients. Because $\mathbf{y}_i$ is determined by $\mathbf{x}_i$, so $\mathbf{y}_i$ can be the representative of $\mathbf{x}_i$. We reshape the vector $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{HWC}]$ into a new image sequence $\mathbf{Y}'$ of D frames of size $H \times W \times C$. The squeezed sequence of D frames can be used as a simplified, yet an information comprehensive representation, that summarizes the dynamics taking place in the given set of K video frames.

3.2.2 Optimization

In this study, the TSP is optimized with respect to the action recognition task. In order to ensure that the projection $\widehat{\mathbf{X}}$ retains as much meaningful spatio-temporal information as possible from the original video sequence $\mathbf{X}$, $\widehat{\mathbf{X}}$ should be close to the original video data $\overline{\mathbf{X}}$. This relies on finding the optimal $\mathbf{A}$ fitting $\mathbf{X}$, aiming to minimize the residuals of projections. Let us denote the mean absolute error (MAE) on projections by l_{proj}, calculated as:

$$l_{proj} = \frac{1}{HWC}\sum_{i}^{HWC} \|\bar{\mathbf{x}}_i - \widehat{\mathbf{x}}_i\|, \tag{3.5}$$

where $\|\cdot\|$ represents the standard L^2 norm in the K-dimensional Euclidean space $\mathbb{R}^K$.

3.2.3 Network Architecture

Temporal Squeeze Network

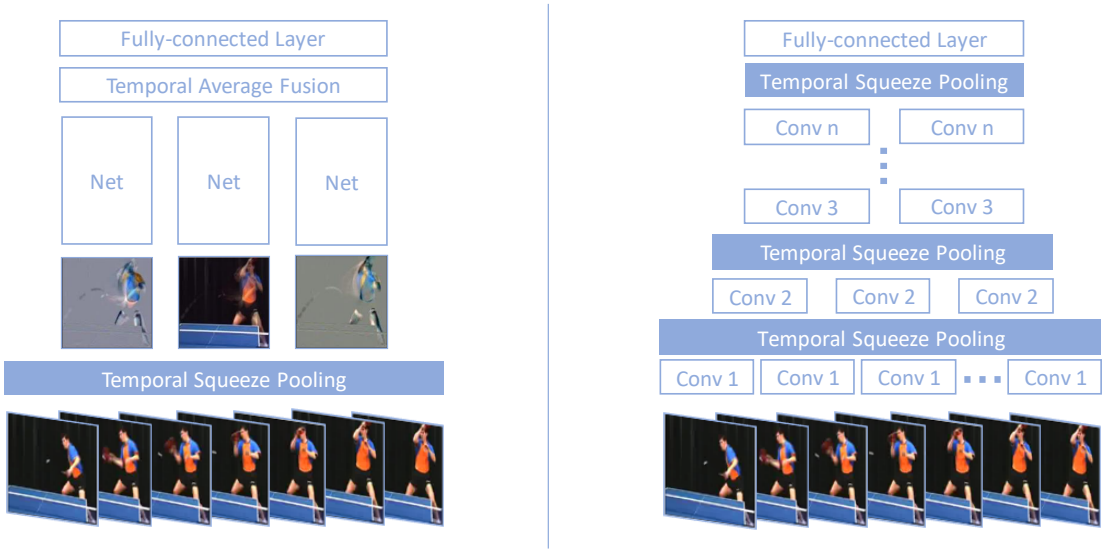
The Temporal Squeeze Pooling can process not just video frames but also the output feature maps of convolutional layers of a CNN. When it is plugged into off-the-shelf CNNs, it forms a new architecture, named Temporal Squeeze Network (TeSNet). We present two different instantiations of TesNet in Fig. 3.3, where structure (a) contain a single TSP layer while structure (b) has multiple TSP layers embedded. Structure (a) can be employed when the input video clip is not too long (*e.g.* K < 16). When the input clip is longer than 16 frames, structure (b) would be preferable, considering that the combination of multiple TSP layers has a stronger information summarization ability than a single TSP layer. Considering that the temporal squeeze pooling module has provided strong motion modeling ability, we prefer employing 2D convolutions rather than 3D convolutions in the network section. Therefore, we can consider using a deeper network than those using 3D convolutions. For example, I3D [16], taken as an example of 3D-CNNs, uses Inception Net [120] as its backbone. Inception-ResNet-V2 is much deeper than Inception Net and has shown remarkable results in many applications. Hence, we choose the Inception-ResNet-V2 [119] as the backbone CNN embedding the TSP layer.

In order to form an end-to-end training, we add the loss term l_{proj} from Eq. (3.5) to the classification loss used in the original network, resulting in the following loss function:

$$l_{final} = l_{classif} + \beta \sum_{i=1}^{M} l^{i}_{proj} + \lambda l_{L2}, \tag{3.6}$$

where $l_{classif}$ is the cross-entropy loss of the classification [119], l_{L2} is the L2 normalization term of all the trainable weights in the architecture, λ is the weight decay, β is the weight for the TSP loss component l_{proj}, where the projection residuals are summed up for all M TSP layers.

TSP layers can be embedded in different locations of the backbone CNN. We design our model by following the principle of decreasing the number of mapped frames D when embedding into a deeper network layer position. In this case, the model represents a pyramidal video processing scheme. The first TSP layer should be configured with a relatively larger D generating more frames, and therefore the loss of temporal information caused by successive pyramidal projections would be reduced.



(a) Embedding a single TSP layer

(b) Several embeddings of TSP layers at various levels of a CNN

Fig. 3.3 Frameworks of two types of Temporal Squeeze Network (TeSNet). The first type of TeSNet (a) has a TSP layer embedded after the input layer for low level feature representation extraction. The second type of TeSNet (b) is configured with multiple TSP layers embedded at different depths in the backbone network.

Two-Stream Model

Inspired by the two-stream architecture [112], we use the optical flow of videos as the second input modality to construct our two-stream model, the framework of which is presented in Fig. 3.4. The spatial TeSNet takes the RGB frames of videos as input, while the temporal TeSNet takes the estimated optical flow of video frames as input. The final prediction of a video is obtained by averaging the output scores of the two streams. In Section 3.3.3, we show that applying a TSP module to the optical flow of videos generates a special movement pattern different from the squeezed images. Moreover, the fusion of two video representation modalities could generate higher accuracy than any single stream.

3.3 Experiments for Temporal Squeeze Pooling

3.3.1 Implementation Details

We conduct experiments on two human activity classification benchmarks, UCF101 [114] and HMDB51 [73]. Each video in the dataset is converted into frames at its original frame

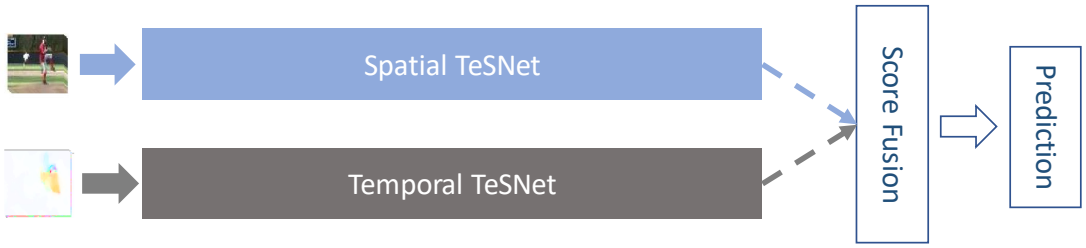


Fig. 3.4 Diagram of the two-stream model, including a spatial TeSNet and a temporal TeSNet.

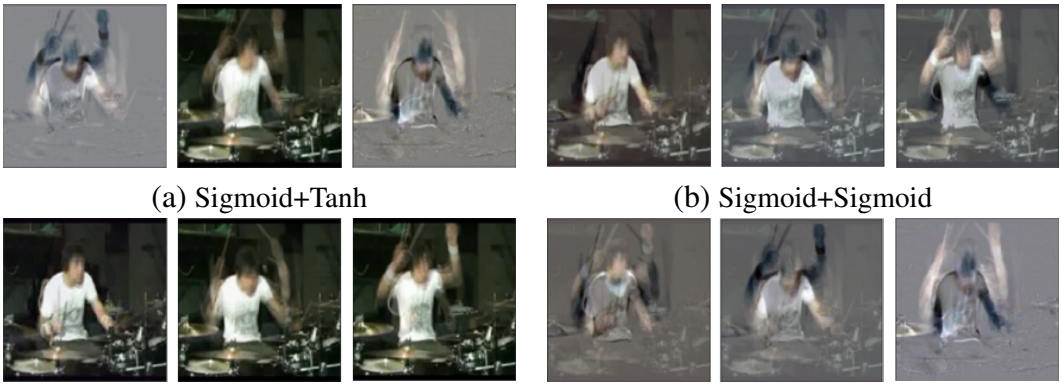
rate. We compute the optical flow using the TVL1 optical flow algorithm [157] and store them as JPEG images after scaling their values to the interval [0, 255]. In our model, we use the RGB images and flow images as input resources for different streams. The spatial size for both RGB images and flow images are 256×340.

Our model is pre-trained on ImageNet [22]. To evaluate our model, we reimplement the Temporal Segment Network (TSN) [143] with our backbone network. We set the dropout as 0.5 to prevent overfitting and adopt the same data augmentation techniques as in [143] for network training. The size of the input frames is set to 299×299, which is randomly cropped from the resized images, and K consecutive frames are randomly selected from each video sequence. We use Stochastic Gradient Descent for optimizing the network parameters in which the batch size is set to 32, momentum of 0.9, weight decay $\lambda = 4e^{-5}$, $\beta = 10$. The initial learning rate is set to 0.001 for the image stream and at 0.005 for the Optical Flow stream. We train the model for 30 epochs, with a ten times reduction for the learning rate when the validation accuracy saturates.

During testing, we uniformly sample 20 clips from each video clip and perform spatially fully convolutional inference for all clips, and the video-level score is obtained by averaging all the clip prediction scores of a video. For the proposed TeSNet, we set $\Phi(\cdot) = I$ in Eq. (3.3), resulting in $\mathbf{A}' = \mathbf{A}$, while the column independent $\mathbf{A}$ is properly initialized. Unless specified otherwise, we consider Sigmoid function for $\delta_1(\cdot)$ and LeakyReLU for $\delta_2(\cdot)$ in Equation (3.2); the temporal squeeze pooling is configured with $K = 10$, $D = 3$, which means the information of a 10-frame video would be summarized into 3 squeezed images; the TSP is placed before the first convolutional layer of the backbone CNN.

3.3.2 Ablation Studies for TSP

Instantiations with various combination of activation functions



(a) Sigmoid+Tanh (b) Sigmoid+Sigmoid

(c) ReLU+ReLU (d) Sigmoid+LeakyReLU

Fig. 3.5 Visualizing the output of the TSP layer when considering different activation functions for δ_1+δ_2 in Eq. (3.2).

In the temporal squeeze pooling, different implementations for activation functions δ_2 and δ_1 from the excitation operation (Eq. (3.2)) have different impacts on the output content of TSP. In order to find out the proper implementations for δ_2 and δ_1, we consider various choices and combinations, commonly used in deep learning, such as: Sigmoid+Tanh where the first corresponds to δ_1 and the second to δ_2, respectively. We train the TSP instances independently with the optimization criteria of Mean Absolute Error (MAE) of the projection on UCF101 (split 1). Numerical and qualitative results for these combinations of activation functions are provided in Table 3.1 and in Fig. 3.5, respectively. From Fig. 3.5a and 3.5d, we can observe that when using Sigmoid+Tanh and Sigmoid+LeakyReLU as the activation functions, the squeezed images achieve a better separation of the moving regions from the background, and they both have smaller MAE, according to the results from Table 3.1. However, when using either Sigmoid+Sigmoid or ReLU+ReLU for δ_1+δ_2, the TSP instance would yield larger MAE, while the moving region and the background are not that well separated, as it can be observed in Fig. 3.5b and 3.5c. In the following experiments we consider δ_1 as Sigmoid and δ_2 as LeakyReLU for instantiating TSP.

Embedding the TSP layer into Networks

In the following, we explore where and how to embed TSP layers into the CNN. The results are shown in Table 3.2, where the second column indicates the location for inserting a TSP

Activation function (δ_1+δ_2)	MAE (l_{proj})
Sigmoid+Tanh	7.14e-2
Tanh+Sigmoid	8.33e-2
Sigmoid+Sigmoid	7.33e-2
ReLU+ReLU	9.86e-2
LeakyReLU+Sigmoid	7.12e-2
Sigmoid+LeakyReLU	**7.01e-2**

Table 3.1 Comparison for the effect of different combinations of types of activation functions for the squeeze function F_{ex} from Eq. (3.2), in terms of MAE of projections. A smaller MAE gives a better result.

layer with the corresponding D indicated in the third column. A single TSP layer, $M = 1$ is embedded in settings No. 1 and 2, while $M = 2$ for settings No. 3, 4 and 5. The model from setting No. 1, which embeds a TSP layer directly after the inputs of the network, achieves the best result in all settings. However, the model with setting No. 5, which embeds two TSP layers into the backbone network, requires fewer computations in FLOPs[1] and has almost the same performance as the No. 1 setting. When a lower level of computational complexity is required, then the No. 5 setting is preferable to be used. Inserting the TSP layers into the middle section of the backbone network leads to worse performance. One possible explanation is that the network was initially pretrained on ImageNet; the inserted TSP layers did not fit well with the settings of these pretrained kernels and resulted in poor performance. In order to avoid this problem, the model with TSP layers embedded has to be pretrained on a large video dataset.

Different clip lengths

We explore how the length of the input video clip affects the performance of our model. we consider a rather small batch size of 8 and a maximum clip length of 64 because of the GPU memory limitation. For the clip length of 64, we adopt the setting No. 5 from Table 3.2 but consider $D_1 = 16$ and $D_2 = 4$. When considering clip lengths of 10 or 16, we use the first setting from Table 3.2. The results are shown in Table 3.3. Due to the small batch size we adopt, we see little performance degradation for the model with a clip length of 10. Nevertheless, it can be observed that when increasing the length of the video clip, the performance improves as well. The model with the clip length of 64 achieves 87.8%

[1] In Inception-ResNet-V2, Block B is located in its latter part. The FLOPs of Setting No. 5 is around 70% of setting No. 1.

No.	Location of TSP layer	Number of squeezed frames(D_i)	Accuracy (%)
1	Input	$D_1 = 3$	**85.4**
2	Input	$D_1 = 1$	83.1
3	Conv2d_1a_3x3 Conv2d_4a_3x3	$D_1 = 3$ $D_2 = 1$	81.7
4	Conv2d_1a_3x3 Block A	$D_1 = 3$ $D_2 = 1$	84.9
5	Conv2d_1a_3x3 Block B	$D_1 = 3$ $D_2 = 1$	<u>85.3</u>

Table 3.2 Evaluating the accuracy when embedding the TSP layer at different depths of the CNN.

Top-1 accuracy on the split 1 of UCF101, which is 2.5% higher than the model with the clip length of 10. The increase in accuracy evidences that our TSP is capable of capturing useful dynamics of very long video sequences.

Clip length	Accuracy (%)
10	85.3
16	86.2
64	**87.8**

Table 3.3 Comparing the effect of various clip length of videos on RGB stream on the split 1 of UCF101 dataset.

Effectiveness of TeSNet

In order to evaluate the effectiveness of the proposed TeSNet architecture, we compare it with the baseline (*i.e*. Inception-ResNet-v2) and TSN [143]. All these models adopt Inception-ResNet-v2 as their backbone networks. The results provided by different architectures and streams are shown in Table 3.4. After employing the proposed TeSNet architecture with the fusion of the RGB and optical flow (OF) modalities, we successfully boosted the Top-1 accuracy from 92.5% to 95.2% on the split 1 of UCF101. TeSNet also outperforms TSN (Inception-ResNet-v2) by 2.3%, which strongly demonstrates the effectiveness of TeSNet.

Architecture	length	RGB	OF	RGB+OF
Baseline	1	83.5	85.4	92.5
TSN	3	85.0	85.1	92.9
TeSNet	64	**87.8**	**88.2**	**95.2**

Table 3.4 Performance of different architectures with two-stream on the split 1 of UCF101 dataset. The baseline is Inception-ResNet-v2, which is also the backbone network of the TSN and TesNet models.

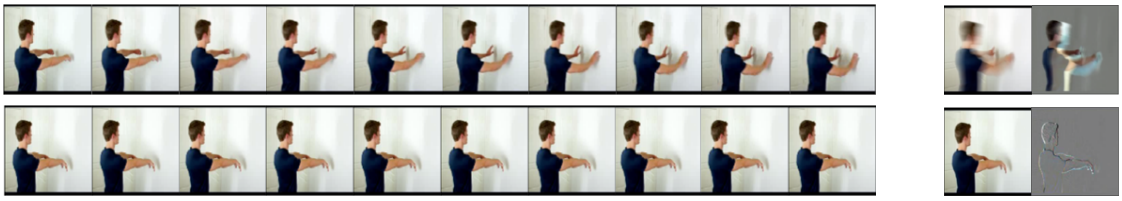
Method	UCF101	HMDB51
iDT+Fisher vector [99]	84.8	57.2
iDT+HSV [98]	87.9	61.1
C3D+iDT+SVM [126]	90.4	-
Two-Stream (fusion by SVM) [112]	88.0	59.4
Two-Stream Fusion+iDT [40]	93.5	69.2
TSN (BN-Inception) [143]	94.2	69.4
Two-Stream I3D [16]	93.4	66.4
TDD+iDT [141]	91.5	65.9
Dynamic Image Network [8]	**95.5**	**72.5**
Temporal Squeeze Network	<u>95.2</u>	<u>71.5</u>

Table 3.5 Temporal Squeeze Network compared with other methods on UCF101 and HMDB51, in terms of top-1 accuracy, averaged over three splits.

3.3.3 Visualization Analysis

We explore how the temporal squeeze pooling represents the spatio-temporal information within the video clips by visualizing its outputs. In Fig 3.6, we show the output of the TSP with $K = 10$, $D = 2$ resulting in 2 squeezed images. The clip, shown on the first row in Fig 3.6a display a clear salient movement, and we can observe that its corresponding squeezed images capture the dynamics of videos, as shown on the first row in Fig 3.6b. The other clip, shown on the second row, does not contain any obvious movement. When there is no movement present in a video clip, the TSP captures the characteristic static information about the scene, as shown in the last two images from the second row of Fig 3.6b.

Fig 3.7 depicts the outputs of the TSP with $K = 10$, $D = 2$. The output of the TSP with RGB frames is shown in Fig. 3.7b, and the output of the TSP that takes optical flow images as input is shown in Fig. 3.7d. We observe that the output of the TSP tends to preserve the static information and the motion information separately. This indicates that by considering



(a) Selection of 10 consecutive frames from video sequence. (b) Squeezed Images

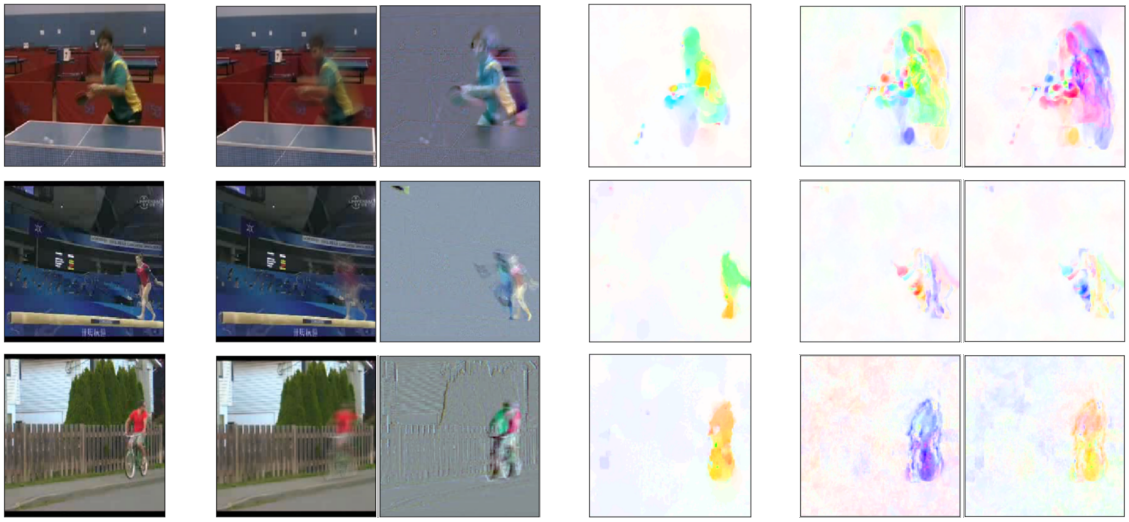
Fig. 3.6 Visualizing the squeezed images in (b) and their corresponding input video clips in (a), where the clip shown on the top row contains obvious movements while the bottom one contains no clear movement. The TSP is configured with $K = 10$, $D = 2$.

a single image we may not be able to represent the underlying spatio-temporal information from the video. Moreover, when considering $D = 3$, the classification accuracy is higher than for $D = 1$, according to the results from Table 3.2. This result further demonstrates that summarizing the dynamics of a long video clip into a single image would lose essential spatio-temporal information. A dynamic image [8] attempts to summarize the entire information from a video clip into a single image, which can explain why they fail to properly represent long video clips.

From the Fig. 3.8, we observe that the squeezed images represent the movement information, depending on the type of movement and moving objects, in different ways. When having moving objects on a stationary background (camera not moving), the background is canceled in the squeezed images, while the moving objects appear as specific blurred regions in the squeezed image, as seen in Fig. 3.8 (a) and (b). When the camera is moving, following a specific fast-moving object, then that object will highlight in the squeezed image, while the moving background will appear as blurred information. In the case when a complex movement of both the multiple moving objects in the scene and the camera following them, then the squeezed image results into a more uncertain representation, as it can be seen from Fig. 3.8 (g) and (h).

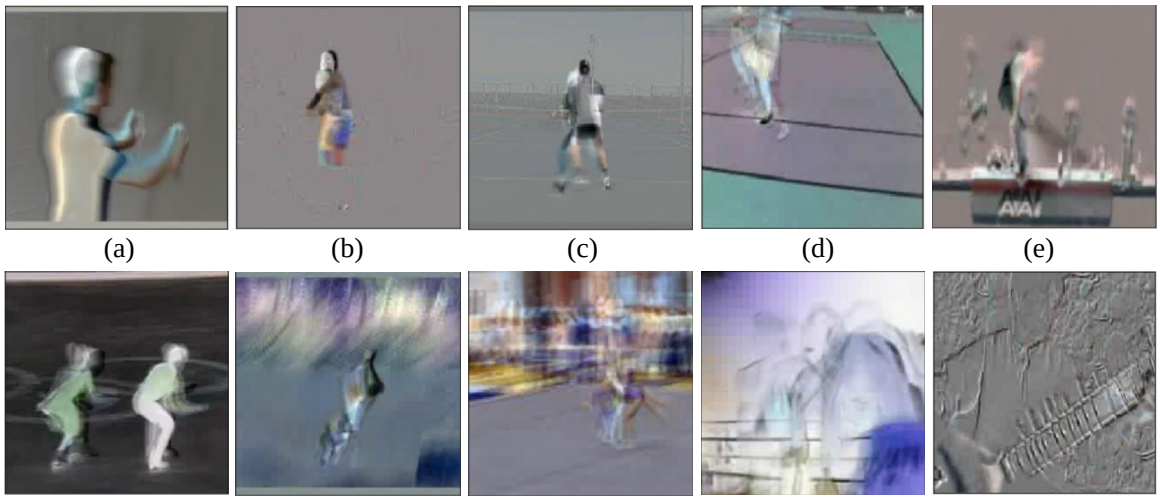
3.3.4 Comparisons

For fair comparisons, we only consider those models that are pre-trained on ImageNet [22]. The results are provided in Table 3.5. The proposed TeSNet achieves 95.2% top-1 accuracy on UCF101 and 71.5% on HMDB51, which outperforms TSN (BN-Inception) by 1% and 2.1% on UCF101 and HMDB51, respectively. As the dynamic image network fuses the prediction scores of four streams using a better backbone network architecture, while our TeSNet only



(a) Single Image (b) Squeezed Image (c) Optical Flow (d) Squeezed Optical Flow

Fig. 3.7 Given input video frames, flow images and the corresponding outputs for the TSP layers ($K = 10$, $D = 2$).



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Fig. 3.8 Examples of squeezed images generated by the temporal squeeze pooling mechanism. In (a)-(f), the moving objects are well captured, as their backgrounds are stationary. In (g)-(i), both object and camera movements are involved, resulting in uncertain representations. In (j), the guitar player does not perform obvious movements, and its squeezed image only presents the textures of the scene.

uses two streams, and therefore the results are not directly comparable. Nevertheless, the advantage of our proposed method is that we can control the number of frames for the output of the TSP layer to preserve the video information better, while the dynamic image method [8] can only summarize a part of the spatio-temporal information into a single image. The proposed TeSNet method can represent the information through TSP from as many as 64 frames, unlike in [8], where the dynamic image method would show performance degradation when processing more than 20 frames.

3.4 Limitations of Temporal Squeeze Pooling

A problem with temporal squeeze pooling is that it would change the temporal order in the movement representation. As a consequence, the model most likely would learn an incorrect temporal relation between video frames. With the development of computer vision, many researchers in the community have started realizing the importance of temporal relations in video sequences for video understanding. Shuffling the order of video frames would not significantly influence the performance of current models on the video datasets that do not require strict temporal order to discriminate different action classes. UCF101 is one of these datasets, so the proposed TSP performs well on this dataset. However, Zhou *et al*. [162] reveal that shuffling the temporal order will cause critical harm to the performance of current models on those video datasets which are defined by the interpretation of the strict temporal order between video frames. So TSP would not work on the "motion-dominant datasets". This fatal shortcoming exists not just in our TSP but also in the rank pooling machines [42, 43, 9, 8], which are used to generate the dynamic images. To address this problem, we can adopt the uniform sampling strategy [143] that uniformly divides a video sequence into multiple segments, and perform the temporal squeeze pooling for each segment, which allows modeling sparse temporal relations of the sampled video frames.

From the visualization analysis in Section 3.3.3, we have demonstrated that the TSP is capable of capturing video dynamics useful for video understanding when videos contain salient movements. However, we find that the TSP is not sensitive to the small local displacement happening between consecutive video frames but is instead sensitive to the illumination change. This is caused by the squeeze operation $F_{sq}(\cdot)$ from Eq. (3.1) implemented as a global pooling, which outputs very close values for the video frames in which movements only happen in some small regions. As a result, the TSP treats the video frames having small local motions equally and fails to represent the characteristic features of these small motions. To address this problem, we need to re-design the temporal squeeze pooling, and the squeeze

operation $F_{sq}(\cdot)$ should be re-defined as a local feature aggregation operation instead of the original global pooling.

3.5 Dynamic Appearance: Video Representation Learning with Joint training

Video data is collected across both space and time, is high-dimensional and contains substantial redundancy. These are challenging to any model aiming to differentiate between relevant and irrelevant spatio-temporal features. While leading-edge CNN architectures [16, 101, 145, 102, 37, 36, 151] have been tailored for learning spatio-temporal features from raw video data, a quintessential spatio-temporal representation remains elusive. In this section, we introduce the concept of Dynamic Appearance, which is a novel video representation summarizing the appearance information relating to movement in a video sequence while filtering out the static information considered unrelated to motion. The filtered static information that does not change from frame to frame is temporarily referred to as the Static Appearance. To better understand the concept, we depict the dynamic appearance and static appearance of a video in Fig. 3.9. Our motivation for this study stems from the observation that many videos labeled with different action categories are shot in the same environment. Taking the videos shown in Fig. 3.9 as an illustrative example, the two videos have different action labels ('Cricket Shot' *vs.* 'Cricket Bowling') but they are actually two short video clips shot in the same environment. In this case, the static appearances and scene information of these videos are very similar, describing the same object and background, which hinders CNNs that take RGB frames as input from learning relevant spatio-temporal features. One solution would be to replace the RGB frame input with a pure motion representation such as optical flow. However, the optical flow would lose appearance information, which could result in misclassifying videos in different scenes. Therefore, the classic two-stream CNN architecture [112] is proposed in order to learn representations of motion and appearance from optical flow and raw RGB frames separately. However, the computational complexity of two-stream-based architecture [112, 38, 40, 143, 16] is double that of the single-stream version.

One question asked in video analysis is: if we can filter out static appearances from video frames and only preserve the dynamic appearances related to the motion, can we circumvent the interference caused by static appearances? To this end, we propose the Pixel-Wise

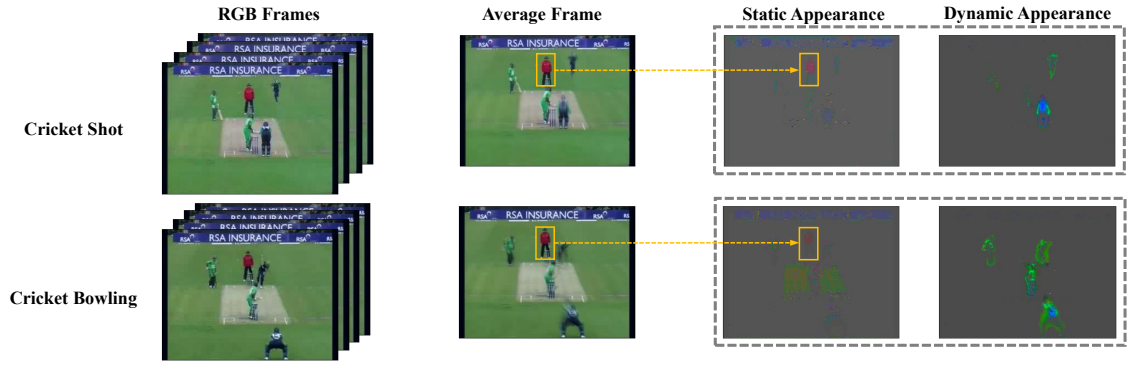


Fig. 3.9 Different actions recorded within the same environment have very similar static appearances, which could cause confusion in classifiers. Using the Dynamic Appearance as an input resource would avoid such confusions, as it only contains the visual information related to the movement. For instance, the person in the bounding box does not move and is not included in the dynamic appearances but rather in the static appearances.

Temporal Projection (PWTP), which is based on the optimization of the projection from a high dimensional space into a lower space, following training on a given video dataset. By utilizing PWTP, the temporal information of a video is projected to a subspace of its original temporal dimension. The resulting projection encodes the static appearance of video while the projection residual encodes the dynamic appearance. As shown in Fig. 3.9, the dynamic appearance manifests motion by encoding only the meaningful visual information related to motion. Hence, CNNs that take dynamic appearances as input could avoid the distraction caused by static appearances and focus on learning high-level spatio-temporal features during training.

PWTP can be regarded as an improved version of Temporal Squeeze Pooling (TSP) described in Section 3.2. The TSP module makes it possible to process very long video clips with a limited computation budget by summarizing the visual information in a long video frame sequence into just a few squeezed images, which preserves the dynamic as well as static information of the video. As discussed in Section 3.4, the squeezed images generated by the TSP module are useful for video recognition provided that there is no severe camera shake or movement. Otherwise, the squeezed images would be blurred, which results in poor discrimination between the moving objects and backgrounds in the scene. This phenomenon will be more severe when processing longer frame sequences. This shortcoming also exist in discriminative rank pooling [41] and dynamic images [8]. The main reason for the poor discrimination is that these pooling methods are intrinsically frame-wise operations that generate an output image by weighted summing up the input frames. However, PWTP

can fundamentally address this problem, as it instead operates at the pixel level, providing different weights for different spatio-temporal positions when pooling along the temporal dimension. Moreover, due to the upgrade from frame level to pixel level, a PWTP module can be utilized to separate the dynamic appearance of a video from its static appearance. In our preliminary work, we experiment that TSP cannot perform the disentangling of dynamic and static appearances. Compared with Squeezed Image [61] and Dynamic Image [8] which summarize both motion and appearance information into one or few images, our dynamic appearance filters out the appearance information unrelated to motion. We demonstrate that dynamic appearance shows higher performance than Squeezed Image and Dynamic Image.

The dynamic appearance learning with PWTP can firstly be regarded as an unsupervised learning method, whose optimization objective is to minimize the projection residual. Alternatively, we can combine the PWTP module with a CNN into a unified learning framework, termed Dynamic Appearance Net (DAN). The dynamic appearance learning works as an auxiliary task to the primary recognition task, and both tasks are optimized with joint training algorithms. It is difficult for a CNN to learn action-relevant features with just the action recognition criteria (*i.e.* cross-entropy loss) due to the presence of irrelevant static appearance features. However, it would be easier for the dynamic appearance learning task. By jointly training the two tasks, the model can better generalize on the action recognition task, as demonstrated in our experiments. We perform extensive experiments on four standard video benchmarks: Kinetics400 [16], Something-Something V1 [48], UCF101 [114] and HMDB51 [73], where we show that PWTP greatly improves the performance in terms of accuracy. The experimental results successfully demonstrate that using the dynamic appearance extracted by PWTP can enable better feature learning in CNNs.

3.5.1 Overview

Motion Representation

Optical flow as a short-term motion representation has been widely used in action recognition. FlowNet series [63, 29] improve the quality of optical flow estimation by using deep learning. Some published works [95, 164] attempt to integrate optical flow estimation and the action recognition model into an end-to-end training framework. However, optical flow represents motion with instantaneous image velocities, which loses the appearance information, and therefore could cause inaccurate classification of videos in different scenes. Derived from the optical flow, OFF [118], Flow-of-Flow [100] and other flow-based methods are proposed for fast motion feature learning. The seminal work [8] utilizes the rank pooling [42] or

approximate rank pooling [8] machine to summarize both the static and dynamic visual information of a video into a few RGB images, called dynamic images. It is worth noting that vanilla PCA cannot achieve the same result as rank pooling [42] because PCA is a generic orthogonalization method for dimension reduction by selecting a set of orthogonal vectors, the output of which would not be adapted to the 2D or 3D layout. SVM pooling [136] generates a video representation similar to dynamic images by exploiting the boundary information of SVM. Nevertheless, dynamic images result in poor discrimination between the moving objects and background, which harms performance when recorded videos are affected by camera shaking. Different from the existing methods, the proposed PWTP achieves the disentanglement of video's static and dynamic appearance. The dynamic appearance can be a preferable input resource to CNNs, encoding the temporal motion information as well as the visual information related to the movement.

Joint Training

Joint Training, one of the guises of multi-task learning, can be regarded as a form of inductive transfer, helping a model generalize better by introducing an inductive bias. Ng *et al*. [95] proposed ActionFlowNet for jointly estimating optical flow and recognizing actions. Similarly, TVNet [33] jointly trained videos' representation learning and the action recognition tasks. In our case, the representation learning for the static and dynamic appearance disentanglement of video is an auxiliary task to action recognition, which is the primary goal. This would allow the model to eavesdrop, *i.e*. learn the motion-relevant representation through the dynamic appearance learning task. Existing methods balance the recognition loss and the objective functions of their representation learning with some fixed scale parameters. In our study, in order to stabilize the feature learning process, we analyze various joint training scenarios, such as the multiple-gradient descent algorithm [23].

3.5.2 Pixel-Wise Temporal Projection (PWTP)

In order to achieve video static and dynamic appearance disentanglement, we propose the Pixel-Wise Temporal Projection (PWTP) operator. Given a T frame video clip $\mathbf{x}$ of spatial resolution $H \times W$, we flatten it along the spatial dimensions, so $\mathbf{x} = \{\mathbf{x}_i\}_{i=1}^{HW}$, $\mathbf{x}_i \in \mathbb{R}^{T \times C}$, and C is the number of channels. The goal of PWTP is to summarize the video information from T frames into D frames and $D << T$. Firstly, for every position i in the spatial dimensions, PWTP learns a particular matrix $\mathbf{A}_i \in \mathbb{R}^{T \times D}$, the column space of which defines a subspace (*i.e*. hyperplane) in $\mathbb{R}^T$. The columns of matrix $\mathbf{A}_i$ are linearly independent. Then the PWTP

projects $\mathbf{x}_i$ onto the corresponding column space of $\mathbf{A}_i$, resulting $\widehat{\mathbf{x}}_i$, given by

$$\begin{aligned} \mathbf{y}_i &= (\mathbf{A}_i^\intercal \mathbf{A}_i)^{-1} \mathbf{A}_i^\intercal \mathbf{x}_i, \\ \widehat{\mathbf{x}}_i &= \mathbf{A}_i \mathbf{y}_i, \quad i = 1, \ldots, HW, \end{aligned} \tag{3.7}$$

where $\mathbf{y} = (\mathbf{y}_1 \ldots \mathbf{y}_{HW})^T$, $\mathbf{y}_i \in \mathbb{R}^{D \times C}$ are the mapping coefficients. The projection result $\widehat{\mathbf{x}}$ preserves the static appearance information shared in T video frames and this is enabled by $\mathbf{y}$.

The production of input-specific $\mathbf{A} = (\mathbf{A}_1 \ldots \mathbf{A}_{HW})$ is realized in two steps: 1) producing a temporal relation descriptor for every spatial position i; 2) a Multilayer Perceptron (MLP) takes as input the temporal relation descriptors and outputs the tensor $\mathbf{A}$. In the first step for the temporal relation description, we pass the input $\mathbf{x}$ to a spatial convolution $\mathcal{F}$ with the kernel size of $k \times k$ and stride of s for local spatial information aggregation:

$$\tilde{\mathbf{x}} = \mathcal{F}_s^{k \times k}(\mathbf{x}; \theta), \tag{3.8}$$

where θ, of size $k \times k \times C \times C'$, denotes the kernel weights. The spatial convolution $\mathcal{F}$ maps the tensor $\mathbf{x}$ to $\tilde{\mathbf{x}}$ of size $\frac{HW}{s^2} \times T \times C'$, which significantly reduces the computational cost for PWTP when setting the stride $s > 1$. This is a non-trivial trick that allows the pixel-wise projection to be implemented within a feasible computation budget. Subsequently, we have $\tilde{\mathbf{x}} = \{\tilde{\mathbf{x}}_i\}_{i=1}^{HW/s^2}$ and $\tilde{\mathbf{x}}_i = \{\tilde{\mathbf{x}}_i^t\}_{t=1}^{T}$, where $\tilde{\mathbf{x}}_i^t \in \mathbb{R}^C$. Then, the temporal relation descriptor $\mathbf{u}_i$ for every spatial position i is obtained as

$$\mathbf{u}_i = \left\{ {}^1\!/_{C'} \tilde{\mathbf{x}}_i^{t_1 \intercal} \tilde{\mathbf{x}}_i^{t_2} \;\middle|\; t_1 \neq t_2, 1 \leqslant t_1, t_2 \leqslant T \right\}. \tag{3.9}$$

Consequently, $\mathbf{u}_i$ is represented as a vector $\mathbf{u}_i \in \mathbb{R}^{\frac{T \times (T-1)}{2}}$ with ascending indices of t_1, t_2, and $\mathbf{u} = [\mathbf{u}_1, \ldots, \mathbf{u}_{\frac{HW}{s^2}}]$. As each element in $\mathbf{u}_i$ describes the similarity between a pair of frames, such that $\mathbf{u}_i$ can represent the relationship between any two frames in the video clip. By taking $\mathbf{u}$ as the input to an encoder, the projection machine would have a better understanding of the temporal relationships in the video, and so provide a better temporal projection mechanism.

The convolution $\mathcal{F}_s^{k \times k}(\cdot; \theta)$ from Eq. (3.8) is important for implementing temporal projection at pixel-level, as it encodes the local spatial information of each position i into the local descriptor $\mathbf{u}_i$, which leads to the projection result $\widehat{\mathbf{x}}$ representing smooth textures. Without Eq. (3.8), the projection in each spatial position would be performed completely indepen-

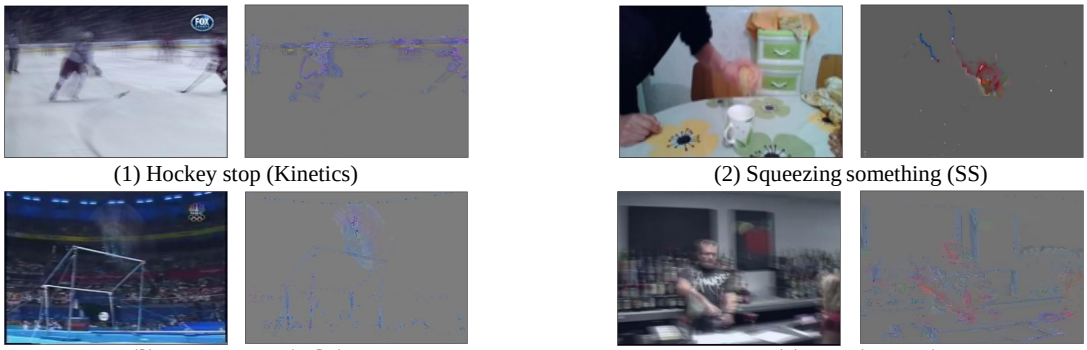
dently, which would fail to consider that the values of neighboring pixels are continuous, resulting in unsmooth spatial textures.

In the second step of the generation of $\mathbf{A}$, we utilize an MLP to process the temporal relation descriptors $\mathbf{u}$ and generate $\mathbf{A}$:

$$\begin{aligned} &\mathbf{A} = \text{Upsample}(\text{MLP}(\mathbf{u})), \\ &\textit{Reshape} : \mathbf{A} \in \mathbb{R}^{HW \times TD} \rightarrow \mathbf{A} \in \mathbb{R}^{HW \times T \times D}, \end{aligned} \tag{3.10}$$

where the 2D bilinear upsampling is employed in the spatial dimensions if convolution $\mathcal{F}$'s stride $s > 1$. Tensor reshaping is then performed at the end. The linear independence of the columns of $\mathbf{A}_i$ can be implemented by properly initializing the MLP (*e.g.* by avoiding to initialize all weights to zero). The MLP in the PWTP module is made up of multiple fully-connected layers with a bottleneck design where the number of features is first reduced by a factor and then recovered to its original size. The residual connection [54] is applied after every bottleneck. Each fully-connected layer, except for the last one, is followed by a GELU non-linearity. More details about the MLP configurations are provided in Section 3.6.3.

Projection Residual as Dynamic Appearance



(1) Hockey stop (Kinetics) (2) Squeezing something (SS)

(3) Uneven Bars (UCF) (4) Pour (HMDB)

Fig. 3.10 Examples of video frame sequences visualized as their frame averaging (left) and corresponding dynamic appearances (right). The videos are randomly picked from Kinetics, Something-Something (SS), UCF101 and HMDB51 datasets.

The projection residual in spatial position i is given by

$$\mathbf{p}_i = \mathbf{x}_i - \widehat{\mathbf{x}}_i. \tag{3.11}$$

Note that when the dimension D of the subspace (*i.e.* rank($\mathbf{A}_i$)) is much smaller than the original temporal dimension T in a video, the projection $\widehat{\mathbf{x}}_i$ is forced to encode the video's static information, because static information repeatedly appear in consecutive frames, construing the main components of $\mathbf{x}_i$. Subtracting the common static features by the original $\mathbf{x}_i$ will generate the dynamic appearance. The projection residual $\mathbf{p}$ preserves the pure dynamic appearance information of the video and filters out the static appearance. Considering that many spatial positions in the projection residual do not carry visual information after optimization, $\mathbf{p}_i \approx \mathbf{0}$, we average $\mathbf{p}$ along the temporal direction to reduce the tensor size:

$$\bar{\mathbf{p}}_i = \frac{1}{T}\sum_{t=1}^{T}\mathbf{p}_i^t. \tag{3.12}$$

Some examples of dynamic appearances for video clips of various activities from Kinetics, Something-Something, UCF101 and HMDB51 datasets are shown in Fig. 3.10.

3.5.3 Dynamic Appearance Net (DAN)

The proposed DAN framework is presented in Fig. 3.11. We sample and then divide the video along the temporal dimension into S segments, with each segment containing T consecutive video frames. The proposed PWTP module is applied to every segment of videos. As a result, the $S \times T$ RGB frames are projected into S frames of dynamic appearance. By taking the dynamic appearance $\bar{\mathbf{p}}$ as input, a CNN can avoid the distraction caused by the static appearance ($\widehat{\mathbf{x}}$) when discriminating the actions in some similar scenes, and therefore can focus on learning high-level motion-relevant features. We simply call the network that takes dynamic appearances as input Dynamic Appearance Net (DAN). In order to evaluate the generalization ability of our methods with different CNN architectures, we consider using TSM R50 [84] and X3D-XS and X3D-M [36] as the backbone networks of our DAN framework. We demonstrate that our DAN improves the performance of these CNNs steadily.

3.5.4 Joint Training

The optimization objective of PWTP is defined to minimize the Euclidean Norm of Projection Residual (ENoPR):

$$\mathcal{L}^1 = \frac{1}{HWC}\sum_{j=1}^{HWC} \| \tilde{\mathbf{p}}_j \|^2, \tag{3.13}$$

where $\tilde{\mathbf{p}} \in \mathbb{R}^{HWC \times T}$ denotes the flattened version of $\mathbf{p}$ along the space and channels. $\mathcal{L}^1$ can be defined as a loss function $\mathcal{L}^1(\Theta^1)$ w.r.t. PWTP's weight parameters Θ^1. For the case of N

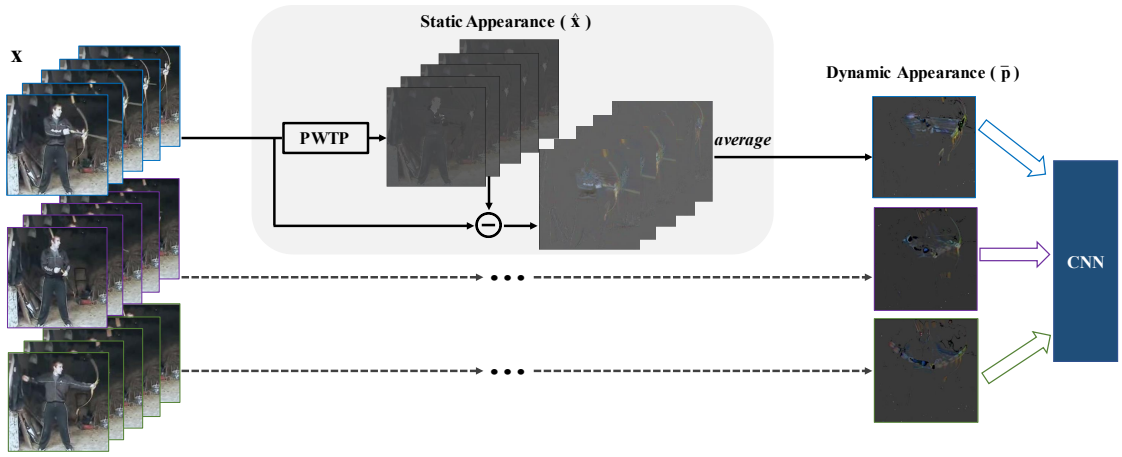


Fig. 3.11 Diagram of the Dynamic Appearance Net (DAN). The Static and Dynamic appearances are separated by the PWTP module. Subsequently, the backbone CNN feeds on the dynamic appearances for learning high-level spatio-temporal features.

data points, the ENoPR is expressed by:

$$\widehat{\mathcal{L}}^1(\Theta^1) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}_n^1(\Theta^1). \tag{3.14}$$

The loss function of action recognition is denoted by $\widehat{\mathcal{L}}^2(\Theta^2)$ in independent training, $\widehat{\mathcal{L}}^2(\Theta^1,\Theta^2)$ in joint training, where PWTP's parameters Θ^1 are shared between the two tasks. The joint training of $\widehat{\mathcal{L}}^1(\Theta^1)$ and $\widehat{\mathcal{L}}^2(\Theta^1,\Theta^2)$ forms the multi-objective optimization :

$$\min_{\Theta^1,\Theta^2} \quad \left[\alpha\widehat{\mathcal{L}}^1(\Theta^1) + (1-\alpha)\widehat{\mathcal{L}}^2(\Theta^1,\Theta^2)\right], \quad 0 \leqslant \alpha \leqslant 1, \tag{3.15}$$

where the weight α, used to scale the contribution of the two loss functions, can be fixed, as in [33], or can be adaptive.

Multiple-Gradient Descent Algorithm (MGDA)

For the adaptive scale α, we adopt the Multiple-Gradient Descent Algorithm (MGDA) [23, 110], in which the goal of multi-objective optimization is to achieve Pareto-optimality. A

solution is said to be of Pareto-stationary if and only if there is $0 \leqslant \alpha \leqslant 1$ such that:

$$\begin{aligned} &\bullet \quad \alpha \nabla_{\Theta^1} \widehat{\mathcal{L}}^1(\Theta^1) + (1-\alpha) \nabla_{\Theta^1} \widehat{\mathcal{L}}^2(\Theta^1, \Theta^2) = 0, \\ &\bullet \quad \nabla_{\Theta^2} \widehat{\mathcal{L}}^2(\Theta^1, \Theta^2) = 0. \end{aligned} \tag{3.16}$$

Pareto-stationarity is a necessary condition for Pareto-optimality. From [23, 110], the decent directions to Pareto-stationarity would be provided by solving the following optimization problem:

$$\begin{aligned} \min_{\alpha} \ \| \alpha \nabla_{\Theta^1} \widehat{\mathcal{L}}^1(\Theta^1) + (1-\alpha) \nabla_{\Theta^1} \widehat{\mathcal{L}}^2(\Theta^1, \Theta^2) \|_2^2, \\ 0 \leqslant \alpha \leqslant 1, \end{aligned} \tag{3.17}$$

which is a one-dimensional quadratic function of α with an analytical solution:

$$\begin{aligned} \alpha' &= \frac{\left(\nabla_{\Theta^1} \widehat{\mathcal{L}}^2(\Theta^1, \Theta^2) - \nabla_{\Theta^1} \widehat{\mathcal{L}}^1(\Theta^1)\right)^{\mathsf{T}} \nabla_{\Theta^1} \widehat{\mathcal{L}}^2(\Theta^1, \Theta^2)}{\| \nabla_{\Theta^1} \widehat{\mathcal{L}}^1(\Theta^1) - \nabla_{\Theta^1} \widehat{\mathcal{L}}^2(\Theta^1, \Theta^2) \|_2^2}, \\ \widehat{\alpha} &= \max(\min(\alpha', 1), 0). \end{aligned} \tag{3.18}$$

Suppose we optimize the objective with mini-batch gradient descent, then the gradient is updated with Algorithm 1:

Algorithm 1: Optimization of the Joint Training

Input: Training tasks: $\widehat{\mathcal{L}}^1(\Theta^1)$, $\widehat{\mathcal{L}}^2(\Theta^1, \Theta^2)$; Learning rate: η;

```
Initialize Θ^1, Θ^2 randomly;
while not converged do
    Compute gradient: ∇_{Θ^1} L̂^1(Θ^1) and ∇_{Θ^1} L̂^2(Θ^1, Θ^2);
    Update α by solving Eq. (3.18);
    Θ^1 = Θ^1 − η ( α ∇_{Θ^1} L̂^1(Θ^1) + (1 − α) ∇_{Θ^1} L̂^2(Θ^1, Θ^2) );
    Θ^2 = Θ^2 − η ∇_{Θ^2} L̂^2(Θ^1, Θ^2);
end
```

Scale Scheduler for Joint training

When using joint training to resolve the multi-objective optimization problem defined in Eq. (3.15), we consider employing a scale scheduler to estimate a dynamic scale α, weighting the contribution of the two components $\widehat{\mathcal{L}}^1(\Theta^1)$ and $\widehat{\mathcal{L}}^2(\Theta^1, \Theta^2)$. The scale scheduler is a hybrid function that initially sets α to a large value and gradually decreases it to a

minimum value, simulating cosine annealing [89]:

$$\alpha = \begin{cases} \frac{1}{2}\left(1+\cos(\pi \log_{\left[(\gamma M)^{\frac{\pi}{\cos^{-1}(2\lambda-1)}}\right]} m)\right), & m < \gamma M \\ \frac{1}{2}\lambda\left(1+\cos(\pi \log_{\left[(1-\gamma)M\right]} m)\right), & m \geq \gamma M \end{cases} \tag{3.19}$$

where m denotes the current iteration, M represents the total number of training iterations, while $\gamma \geq 1$ and $\lambda \leq 1$ are two additional parameters used to control the graph of the hybrid function. Some example graphs of the hybrid function are displayed in Fig. 3.12. The scale

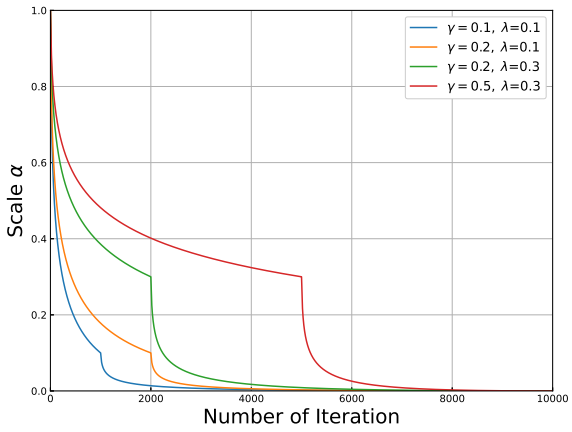


Fig. 3.12 Example graphs of the scale schedule function with different γ and λ.

scheduler is established under the realistic assumption that in the early training period, the feature representation learning task $\widehat{\mathcal{L}}^1(\Theta^1)$ has higher priority than the recognition task $\widehat{\mathcal{L}}^2(\Theta^1,\Theta^2)$. In the later training period, the primary action recognition task should be the main focus, and the optimization objective of the feature representation learning task should be assigned a very low weight.

3.6 Experiments for Pixel-Wise Temporal Projection

We conduct experiments on five action recognition datasets, including Kinetics400 [16], Mini-Kinetics (Mini-K) [151], UCF101 [114], HMDB51 [73] and Something-Something-V1 [48]. Mini-K is sub-set of Kinetics400 with 200 categories. Some ablation studies are experimented on Mini-K for fast exploration. In Kinetics400, Mini-K, UCF101 and HMDB51, temporal relation is less important than the RGB scene information.

In the following, we first describe the implementation details. Then we provide the main results on multiple standard video benchmarks, and make comprehensive comparisons with the state-of-the-art methods. To demonstrate the scientificity and rationality of our methods, we conduct ablation studies to analyze each component of the method individually.

3.6.1 Implementation Details

Unless specified otherwise, our models adopt the following settings: the PWTP module embedded in DAN is configured with $T = 8, D = 1$. The convolution $\mathcal{F}_s^{k\times k}(\cdot;\theta)$ is configured with the kernel size of 9×9, stride $s = 8$ and $C' = 24$ output channels. For the models on Something-Something V1, UCF101 and HMDB51, we use the uniform sampling strategy in TSN [143], where a video is evenly divided into S segments. For each segment, we select T consecutive video frames to form a clip of $S \times T$ frames. For the models on Kinetics400 [16], we perform dense sampling and select $S \times T$ consecutive frames. The dynamic appearance extraction is performed on every T consecutive frames.

We train our models using synchronized SGD with momentum 0.9 and a cosine learning rate schedule. Following the experimental settings in [84, 143], the learning rate and weight decay parameters for the classification layers are 5 times larger than those for the convolutional layers. In order to alleviate over-fitting, we apply L2 regularization to the convolutional and classification layers, and a dropout layer with the ratio of 0.5 is added before the classification layer of the DAN model. During training, we apply random horizontal flip as data augmentation to the training data except for Something-Something. We jointly train the dynamic appearance learning and action recognition tasks with the Multiple-Gradient Descent Algorithm described in Section 3.5.4, unless specified otherwise. When using X3D-XS [36] as the backbone, we randomly crop 160×160 pixels from a video with a shorter side randomly sampled within [182, 228] pixels, as in [36]. Otherwise, we randomly crop 224×224 pixels from a video with the shorter side randomly sampled within [256, 320] pixels.

For efficient inference, we sample a single clip per video with center cropping used in the ablation studies. The spatial size of a center crop is of 160×160 for the models with X3D-XS backbone, while for the others is 224×224. When pursuing high accuracy, we sample multiple clips&crops from the video and average the Softmax scores of multiple spacetime "views" (spatial crops$\times$temporal clips) for prediction. Following the practice in [37], we approximate the fully-convolutional testing by taking 3 crops of 256×256 pixels (160×160 for X3D-XS) to cover the spatial dimensions.

Hyperparameters for the DAN model based on ResNet

The backbone networks are pretrained on ImageNet [22]. On Kinetics400, we train our models on 64 GPUs (NVIDIA Tesla V100). The initial learning rate, batch size, total epochs, weight decay and dropout ratio are set to 0.08, 512 (8 samples per GPU), 100, 2e-4 and 0.5, respectively. On Something-Something V1, we train our models in 32 GPUs, and the hyperparameters mentioned above are set to 0.12, 256 (8 samples per GPU), 50, 8e-4 and 0.8, respectively. We use linear warm-up [89] for the first 7 epochs to overcome early optimization difficulty. When fine-tuning the Kinetics models on UCF101 and HMDB51, we train our models on 16 GPUs and freeze the entire batch normalization [65] layers except for the first one to avoid overfitting, following the recipe in [143]. The initial learning rate, batch size, total epochs, weight decay and dropout ratio are set to 0.001, 64 (4 samples per GPU), 10, 1e-4 and 0.8, respectively.

Table 3.6 Results on Something-Something V1. "N/A" indicates the numbers are not available. "Frames" indicates the frames of a given input modality. "views" indicates spatial crops × temporal clips. † denotes our reimplementation.

Method	**Modality**	**Pre-train**	**Backbone**	**Frames ×views**	**FLOPs**	**#Param.**	**Top-1 (%)**	**Top-5 (%)**
TRN [162]	RGB+Flow	ImageNet	BNInc.	(8+40)×1	N/A	36.6M	42.0	-
TEA [81]			R50	16×30	70G×30	24.4M	<u>52.3</u>	<u>81.9</u>
ir-CSN [127]	RGB	None	3D R152	32×10	74G×10	29.7M	49.3	-
NL I3D [145]		Kinetics	3D R50	32×2	168G×2	N/A	44.4	76.0
TSM [84]	RGB	ImageNet	R50	8×6	42.9G×6	23.8M	48.7	77.9
TSM [84]	Flow		R50	40 ×6	N/A	48.6M	39.5	70.5
TSM [84]	RGB+Flow		R50	(8+40)×3	N/A	48.6M	50.6	80.1
DAN TSM$_{S=8,T=4}$	DA		TSM R50	8×6	43.1G×6	23.8M	50.1	78.6
X3D-XS† [36]	RGB	Kinetics	-	4×6	0.6G×6	3.33M	40.6	70.5
DAN-XS$_{S=4,T=8}$	DA		X3D-XS	4×6	0.8G×6	3.34M	43.1	73.4
X3D-M† [36]	RGB		-	16×6	6.3G×6	3.33M	52.0	81.0
DAN-M$_{S=16,T=3}$	DA		X3D-M	16×6	7.0G×6	3.34M	**53.7**	**82.1**

Hyperparameters for the DAN model based on X3D

On Kinetics400, we train our models on 32 GPUs. The initial learning rate, batch size, total epochs, weight decay and dropout ratio are set to 0.4, 256 (8 samples per GPU), 256, 5e-5 and 0.5, respectively. On Something-Something V1, the backbone networks are pretrained

on Kinetics. We train our models on 32 GPUs, and these hyperparameters are set to 0.12, 256 (8 samples per GPU), 50, 4e-4 and 0.8, respectively.

3.6.2 Main Results

Results on Something-Something

Table 3.6 summarizes a comprehensive comparison, including the inference protocols, corresponding computational costs (FLOPs) and the prediction accuracy. The data sampling details (*i.e*. S, T) of our models are provided in the subscripts beside the model names. Our method surpasses the listed methods by good margins. Our DAN TSM taking dynamic appearance as input has 1.4% and 10.6% higher top-1 accuracy than TSM R50 taking RGB frame as input and TSM R50 taking optical flow as input, respectively. Moreover, our DAN TSM achieves similar accuracy (50.1% *vs*. 50.6%) to the ensemble model of TSM that fuses the prediction of the RGB frame and optical flow modalities. Benefiting from the lightweight and low

Table 3.7 Results on Kinetics400. We report the inference cost of multiple "views" (spatial crops × temporal clips).

Method	**Backbone**	**Frames** × **views**	**FLOPs**	**Top-1 (%)**	**Top-5 (%)**
bLVNet-TAM [34]	bLR50	16×9	561G	72.0	90.6
STM [68]	R50	16×30	2010G	73.7	91.6
SlowFast$_{4\times16}$ [37]	3D R50	32×30	1083G	75.6	92.1
ip-CSN [127]	3D R101	32×30	2490G	76.8	92.5
SmallBigNet [80]	R101	32×12	6552G	<u>77.4</u>	**93.3**
I3D$_{\text{RGB}}$ [16]	Inc. V1	64×N/A	N/A	71.1	89.3
NL I3D [145]	3D R101	128×30	10770G	**77.7**	**93.3**
TSM [84]	R50	16×30	2577G	74.7	-
DAN TSM$_{S=16,T=8}$	TSM R50	16×30	2602G	75.9	92.2
X3D-M [36]	-	16×30	191G	76.0	92.3
DAN-M$_{S=16,T=8}$	X3D-M	16×30	216G	77.3	<u>92.8</u>

parameter redundancy advantages of X3D architecture [36], our DAN-XS with X3D-XS backbone produces a similar accuracy as NL I3D [162], but requires 210 times fewer FLOPs per spacetime 'view'. Meanwhile, DAN-XS also produces higher top-1 accuracy (+2.5%) than X3D-XS. By adopting X3D-M as the backbone, our DAN-M achieves 53.7% top-1 accuracy, which is 1.7% higher than X3D-M and 1.4% higher than TEA [81]. Notably, our

Table 3.8 Results on HMDB51 and UCF101. We report the mean class accuracy (%) over the three official splits.

Method	Backbone	HMDB51	UCF101
StNet [53]	R50	-	93.5
TSM [84]	R50	73.5	95.9
STM [68]	R50	72.2	96.2
TEA [81]	R50	73.3	**96.9**
DI Four-Stream [8]	ResNeXt101	72.5	95.5
TVNet [33]	BNInception	71.0	94.5
$TSN_{RGB+Flow}$ [143]	BNInception	68.5	94.0
$OFF_{RGB+Flow}$ [118]	BNInception	<u>74.2</u>	96.0
DAN TSM	TSM R50	**75.3**	<u>96.5</u>

DAN architecture can improve the performance of existing spatio-temporal CNNs such as TSM and X3D by simply employing them as its backbone networks, which indicates that our method can improve generalization abilities in CNNs.

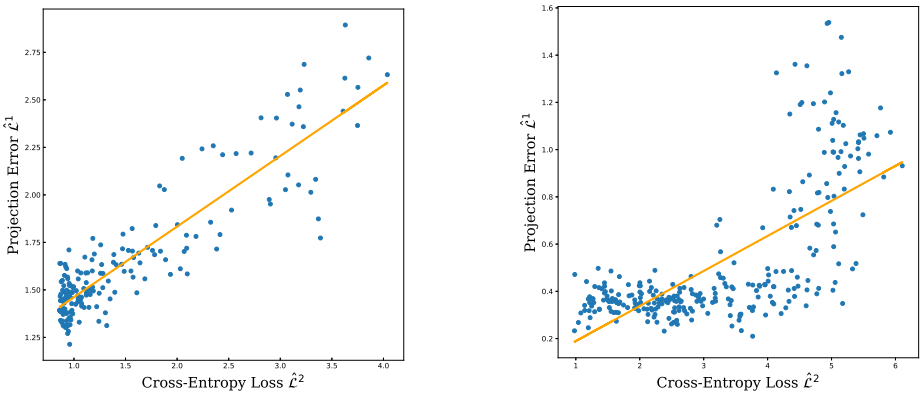
Results on Kinetics400, UCF101 and HMDB51

Table 3.7 shows the results on Kinetics400, where we list the models with the spatial input size of 256^2. DAN-M achieves 77.3% top-1 accuracy, which is better than X3D-M by 1.3% and better than I3D [16] by a margin of +6.2%. By either adopting X3D-M or TSM R50 as the backbone, our DAN method can consistently improve the performance, which firmly demonstrates its effectiveness. Although the scene information on Kinetics is widely considered to be important when distinguishing some actions, the improvement caused by filtering out the static appearance suggests that static features that do not change from frame to frame may impede the enhanced feature learning abilities in standard networks. The results on UCF101 [114] and HMDB51 [73] are shown in Table 3.8, where we report the mean class accuracy over the three official splits. The recognition accuracy is already saturated on these relatively small-scale datasets. In order to avoid over-fitting, we pre-train our model on Kinetics. We consider the inference protocol (3 crops$\times$2 clips) for accuracy. DAN TSM outperforms most other methods without employing tricks such as multi-stream fusion.

3.6.3 Ablation Studies for PWTP

In this section, we conduct ablation studies on different components of the proposed PWTP to prove the scientificity and rationality of our model design. To investigate more configurations, we employ the smaller X3D-XS [36] as the backbone network processing $S = 4$ segments, unless specified otherwise.

Smaller Projection Error, Higher Classification Accuracy?



(a) UCF101 (b) Mini-Kinetics

Fig. 3.13 Example scatterplots of variables $\widehat{\mathcal{L}}^1$ and $\widehat{\mathcal{L}}^2$ on dataset (a) UCF101 and (b) Mini-Kinetics.

The criterion of the dynamic appearance learning task $\widehat{\mathcal{L}}^1(\Theta^1)$ and the action recognition criterion $\widehat{\mathcal{L}}^2(\Theta^1,\Theta^2)$ are supposed to be positive within a certain range, but this is not always true. To find out the correlation of $\widehat{\mathcal{L}}^1$ and $\widehat{\mathcal{L}}^2$ we optimize the model with only $\widehat{\mathcal{L}}^2$ and record the changes for $\widehat{\mathcal{L}}^1$ and $\widehat{\mathcal{L}}^2$ in different training periods. From the plots in Fig. 3.13(a) and (b) on UCF101 and Mini-Kinetics, respectively, we observe that $\widehat{\mathcal{L}}^1$ and $\widehat{\mathcal{L}}^2$ show positive correlation within the chosen range. The positive correlation indicates that it is feasible to set the projection task $\widehat{\mathcal{L}}^1(\Theta^1)$ of PWTP as an auxiliary task helping the model to generalize better on the action recognition task $\widehat{\mathcal{L}}^2(\Theta^1,\Theta^2)$. Outside of the specific range, we may obtain a negative correlation, given the evidence from Table 3.9, where separate training causes $\widehat{\mathcal{L}}^1$ to over-focus on the auxiliary task, resulting in a performance drop on the primary recognition task. When optimizing the PWTP module independently, a too small $\widehat{\mathcal{L}}^1$ could result in PWTP summarizing both static and dynamic appearance information into the same subspace. Meanwhile, the projection residual $\mathbf{p}$ would carry very little information, failing to represent abundant dynamic appearances. To tackle this issue, we set $D << T$, as stated in Section 3.5.2.

Joint training

The dynamic appearance learning task and action recognition task could either be optimized separately or through joint training. In Table 3.9 we investigate the effect of separate training and different joint training approaches on performance. The scale scheduler works better than the fixed scale method. Although the scale scheduler introduces two additional parameters λ and γ, its performance is not that sensitive to changes in these parameters. In summary, joint training approaches have higher accuracy than the separate training strategy. We note that the joint training with MGDA [23] produces higher accuracy than the fixed scale strategy, which demonstrates the effectiveness of MGDA.

Table 3.9 Results for different joint training approaches on Something-Something V1. The baseline is the network with RGB frame input.

Joint Training Approach	ENoPR ($\widehat{\mathcal{L}}^1$)	Accuracy (%)
Baseline	-	38.3
Separate training	0.17	39.4
Constant Scale ($\alpha = 0$)	0.43	39.2
Constant Scale ($\alpha = 0.5$)	0.34	40.1
Constant Scale ($\alpha = 0.9$)	0.22	39.3
Scale Scheduler ($\gamma = 0.1, \lambda = 0.1$)	0.57	40.7
Scale Scheduler ($\gamma = 0.2, \lambda = 0.3$)	0.56	40.7
Scale Scheduler ($\gamma = 0.2, \lambda = 0.3$)	0.56	40.7
Scale Scheduler ($\gamma = 0.5, \lambda = 0.3$)	0.55	40.4
MGDA	0.27	**41.0**

Instantiations and Settings of PWTP

Table 3.10 Evaluating PWTP when changing D and T. The computational cost (FLOPs) of X3D-XS with a PWTP module embedded is reported.

T	D	Mini-K		SS V1		FLOPs
		ENoPR	Accuracy	ENoPR	Accuracy	
4	1	0.26	56.2	0.16	39.3	0.71G
4	2	0.24	55.3	0.08	39.5	0.71G
8	1	0.41	56.9	0.27	41.0	0.82G
8	3	0.33	<u>57.0</u>	0.13	<u>41.3</u>	0.83G
16	1	0.72	**58.6**	0.39	**42.1**	1.12G

We consider various configurations of PWTP on SS V1 and Mini-Kinetics, vary T and D from Eq. (3.10) and evaluate the recognition accuracy. Table 3.10 shows that the configurations with $T = 8$ have higher accuracy than $T = 4$ but lower accuracy than $T = 16$, which suggests that longer frame sequences can help PWTP to capture a diversity of dynamic appearances regarding movement and generate better results. When fixing $T = 8$, the setting of $D = 3$ is better than that of $D = 1$, which suggests that a relatively larger D endues PWTP with stronger information summarization capacity and helps generate lower ENoPR and higher accuracy. Nevertheless, D should not be larger than half of T, as we have seen a performance drop when setting $D > \frac{1}{2}T$ in our preliminary experiments.

The spatial convolution $\mathcal{F}_s^{k\times k}(\cdot;\theta)$ from Eq. (3.8) is used for local information aggregation. We ablate various settings of $\mathcal{F}_s^{k\times k}(\cdot;\theta)$, including changing the stride s, kernel size $k \times k$, output channel number C' and the results are provided in Table 3.11. A larger kernel of $\mathcal{F}_s^{k\times k}(\cdot;\theta)$ can receive information from a larger local region, and therefore PWTP can encode additional spatial information into the temporal descriptor $\mathbf{u}$, which helps the model to produce higher accuracy. In addition, Table 3.11 shows that a large stride s for the convolution $\mathcal{F}_s^{k\times k}(\cdot;\theta)$ can significantly reduce the computational cost while maintaining performance.

Table 3.11 Evaluating various settings of convolution $\mathcal{F}_s^{k\times k}$ in the PWTP module. The computational cost (FLOPs) of X3D-XS with a PWTP module embedded is reported.

Stride	Kernel size	#Channels	Mini-K		SS V1		FLOPs
s	$k \times k$	C'	ENoPR	Accuracy	ENoPR	Accuracy	
1	7×7	12	0.26	55.1	0.14	40.9	2.60G
1	7×7	24	0.22	**58.6**	0.13	41.1	4.06G
4	7×7	24	0.32	58.1	0.18	**41.3**	0.95G
8	9×9	24	0.41	56.9	0.27	41.0	**0.82G**
8	11×11	24	0.37	57.1	0.20	41.2	0.86G

Efficiency and Effectiveness of the Dynamic Appearance

We draw an apple-to-apple comparison between the proposed dynamic appearance and other motion representations [8, 33, 63, 157]. The comparison results are shown in Table 3.12. For all the video representation methods, we guarantee they consume the same number of raw video frames (48). In order to fairly evaluate different video representation methods [8, 33,

Table 3.12 DA *vs*. other motion representation methods. † denotes our reimplementation. The additional parameters and computation (FLOPs) required by the representation methods are reported.

Representation Method	Efficiency Metrics		SS V1	UCF101
	FLOPs	#Param.		
RGB	-	-	46.5	87.1
RGB Difference [143]	-	-	46.6	87.0
TV-L1 Flow [157]	-	-	37.4	88.5
RGB+Flow	-	-	**49.8**	**93.9**
Dynamic Image† [8]	-	-	43.3	86.2
FlowNetC† [63]	444G	39.2M	26.3	87.3
FlowNetS† [63]	356G	38.7M	23.4	86.8
TVNet† [33]	3.30G	0.20K	45.2	88.6
Squeezed Image	-	-	42.7	88.5
DA	0.23G	7.13K	<u>48.7</u>	<u>89.7</u>

63, 157, 61], we reimplement these methods in the same experimental settings with the code provided by the original authors. In our reimplementation, Dynamic Image and Squeezed Image methods generate one frame of representation for every 6 video frames, which is the same as our method. As for TVNet and TV-L1 Flow, we stack 5 frames of the estimated flow along the channel dimension to form an input frame with 10 channels, in total processing 6 RGB frames in a segment. All the models are pretrained on ImageNet and then trained with the same hyperparameters. The motion representations produced by these methods are used as inputs to the network TSM R50 [84]. The prediction scores are obtained by the average consensus of eight temporal segments [143]. The network that takes the proposed dynamic appearance as input outperforms all other motion representation methods by big margins within reasonable computational costs. In SS V1, our method achieves similar accuracy to the fusion of "RGB+Flow".

Design of Multilayer Perceptron (MLP)

The MLP in the PWTP module is used to generate the input-specific $\mathbf{A}$ in Eq. (3.10). In Fig. 3.14, we present the scheme of the MLP, where the feature numbers of the fully-connected (FC) layers are specified on their right side. The first FC expands the number of features with a factor of r. The MLP contains B bottleneck blocks. Each bottleneck block comprises a normalization layer and two FC layers, where the number of features is first

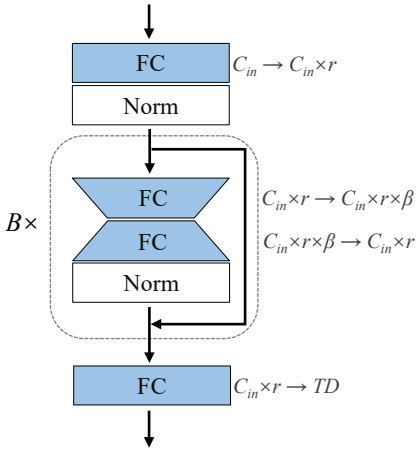


Fig. 3.14 The scheme of MLP in the PWTP module.

Config	Bottleneck ratio (β)	Number of blocks (B)	Expansion ratio (r)	ENoPR ($\widehat{\mathcal{L}}^1$)	FLOPs	Params
#1	1/4	1	4	0.26	45.3M	16.9k
#2	1/4	1	1	0.27	38.0M	7.4k
#3	1/4	2	1	0.28	38.3M	7.9k
#4	1/4	2	4	0.26	50.2M	23.3k
#5	1	1	4	0.28	59.8M	35.5k
#6	1/4	0	4	0.30	40.0M	10.0k
#7	1/4	1	2	0.27	39.7M	9.7k

Table 3.13 Various configurations of the MLP (Lower ENoPR ($\widehat{\mathcal{L}}^1$) means higher capacity for representation). We report the computational cost (FLOPs) of the PWTP module.

reduced by a factor of β and then recovered to its original size. For the normalization, we use Batch Normalization, which behaves more stably than Layer Normalization in our study. In Eq. (3.10), the number of input channels for the MLP is $C_{in} = \frac{T\times(T-1)}{2}$, while the number of output channels is TD. Here, we treat PWTP as an individual optimization problem and evaluate its representation capacity by ENoPR $\widehat{\mathcal{L}}^1(\Theta^1)$ on Mini-Kinetics. The specifications of the MLP's various configurations are listed in Table 3.13, in which the PWTP modules are trained for 10 epochs with AdamW optimizer [90]. We can see that aside from Config #6, which does not use bottleneck blocks, the rest of the configurations have similar ENoPR ($\widehat{\mathcal{L}}^1$). To balance the capacity/speed trade-off, we use Config #7 as the default MLP configuration in our experiments.

3.6.4 Visualization Analysis

In Figure 3.10, we provide four example videos of single segments and their corresponding dynamic appearances. The dynamic appearances are stable in the cases of jittering and other camera movements. For example, the video in Fig. 3.10(4), describing the pouring action, contains significant camera movement, but its dynamic appearance suppresses the background movement and stationary information and retains the appearance information characteristic to motion.

Dynamic Appearance of longer frame sequences

Fig. 3.15 Example videos of different lengths T, visualized as their frame averaging and corresponding dynamic appearances.

We attempt to extract the dynamic appearances from video frame sequences of different lengths T. Some visualization examples are presented in Fig. 3.15. We can observe that the dynamic appearances generated by longer frame sequences contain additional visual information of the movement. According to the results shown in Table 3.10, we observe that by using longer frame sequences to generate dynamic appearances, we get higher accuracy. This observation shows that the movement's additional appearance information captured by PWTP contributes to the performance gain.

Visual Comparison with other Representations

In Fig. 3.16 we compare dynamic appearances with TV-L1 Flow [157] and TVNet [33]. TV-L1 Flow and TVNet represent motion with instantaneous image velocities, so we cannot

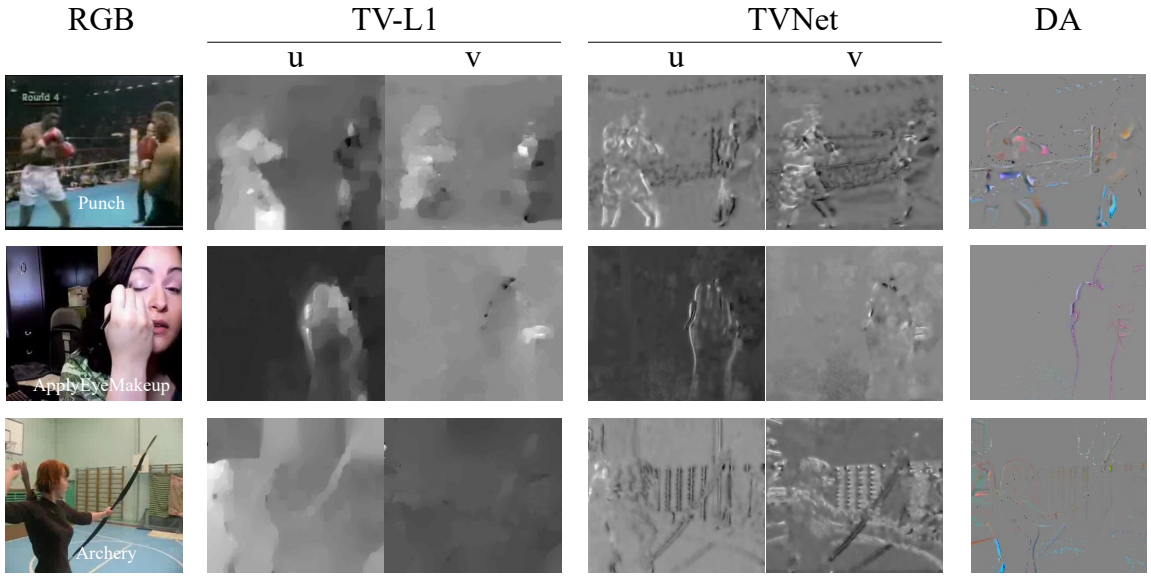


Fig. 3.16 Visual examples of different video representations. Best viewed in color and zoomed in.

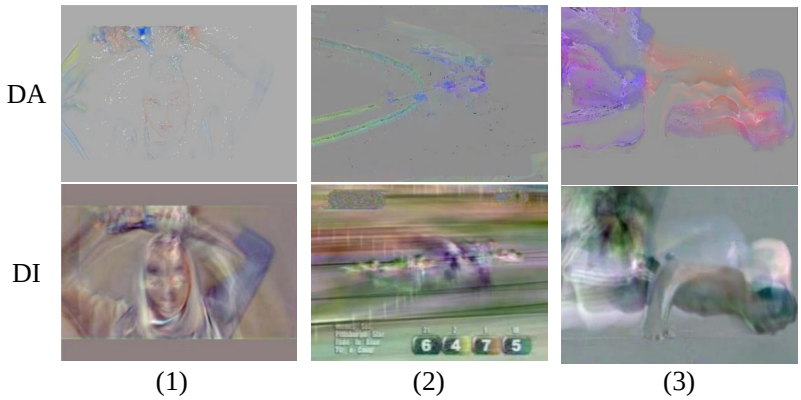


Fig. 3.17 Visualization of Dynamic Appearance (DA) and Dynamic Image (DI). To directly compare DI with our DA visually, we reuse the examples from [8]. The dynamic appearances are generated by consuming $T = 16$ consecutive RGB frames.

observe the appearance information related to moving objects. However, the dynamic appearances preserved the essential parts of moving objects' visual information, which is vital for discriminating different actions. In Fig. 3.17, we compare visually the Dynamic Appearance and Dynamic Image [8]. A dynamic image is essentially a weighted average of multiple video frames, representing the dynamic information of a video by using the approximate rank pooling [8]. Similarly, Squeezed Image archives the same purpose by using the temporal squeeze pooling, described in Section 3.2. However, as we can observe from Fig. 3.17(2), the dynamic image results in poor discrimination between the moving objects and background when we have camera shaking or movement. On the contrary, the dynamic appearance is robust to camera movement, clearly representing the visual information related to motion.

More Visualization Examples

We provide more visualization examples in Figures 3.18 and 3.19, where we provide the averaging of sets of frames from video clips and underneath their corresponding dynamic appearances. From these results, containing a wide range of videos, with 2 examples from each of UCF, Something Something V1 (SS V1), Kinetics and HMDB datasets, we can observe that the dynamic appearance extracted PWTP retains the essence of movement for a wide range of actions.

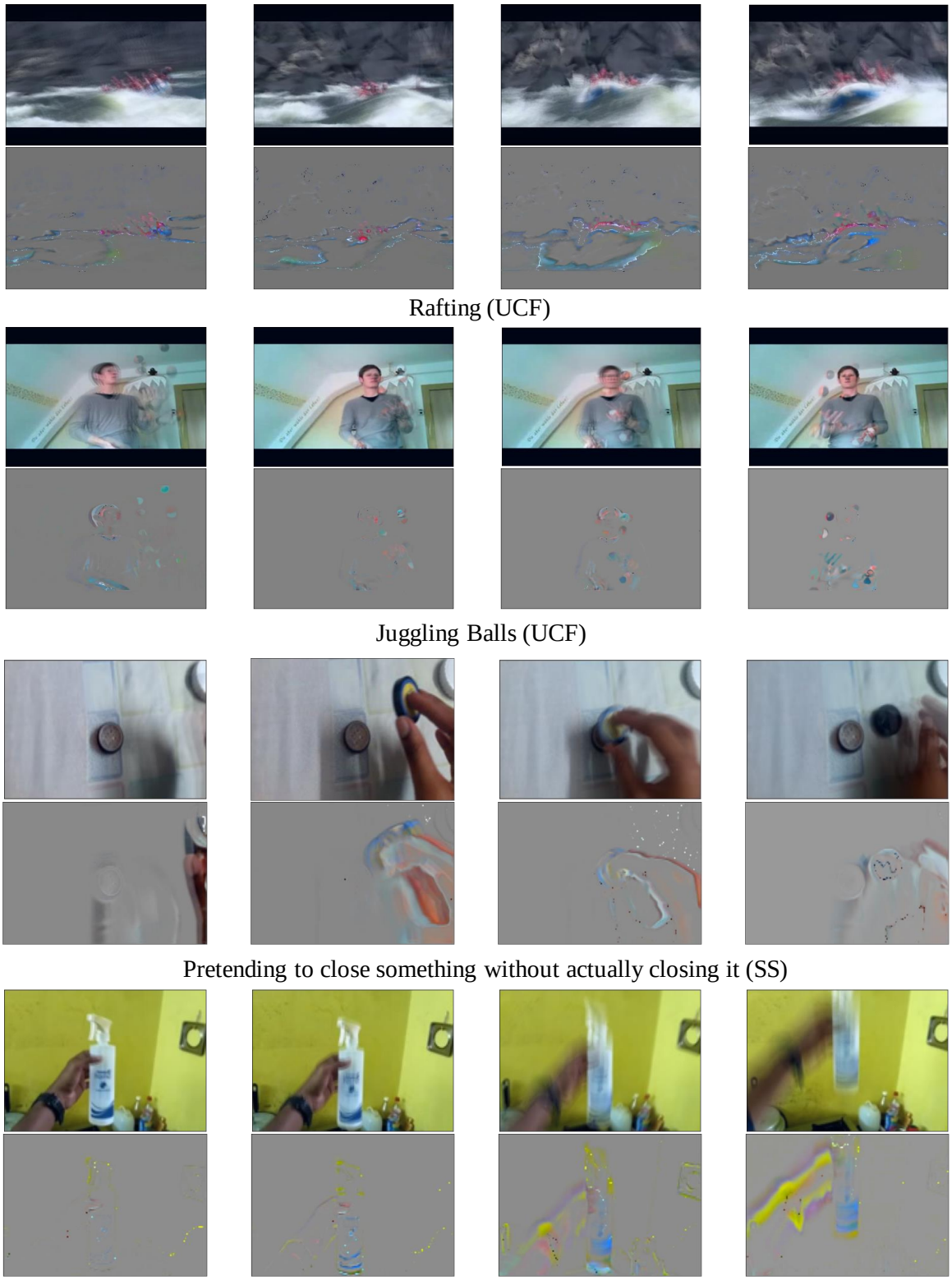
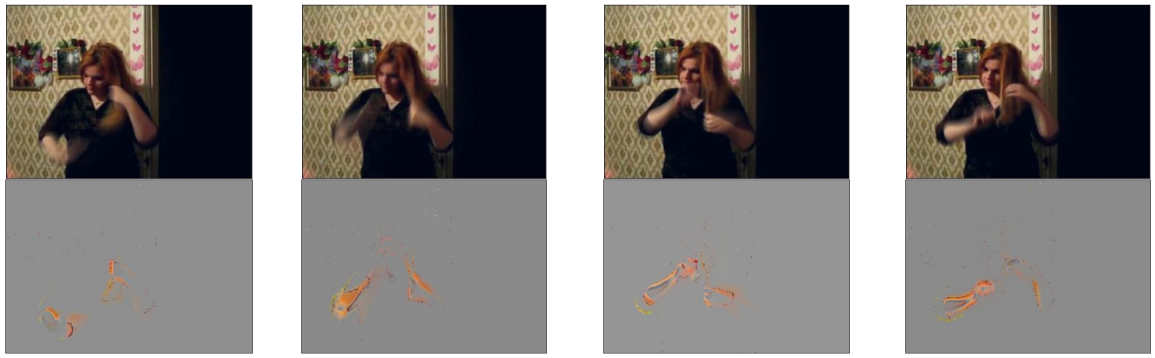


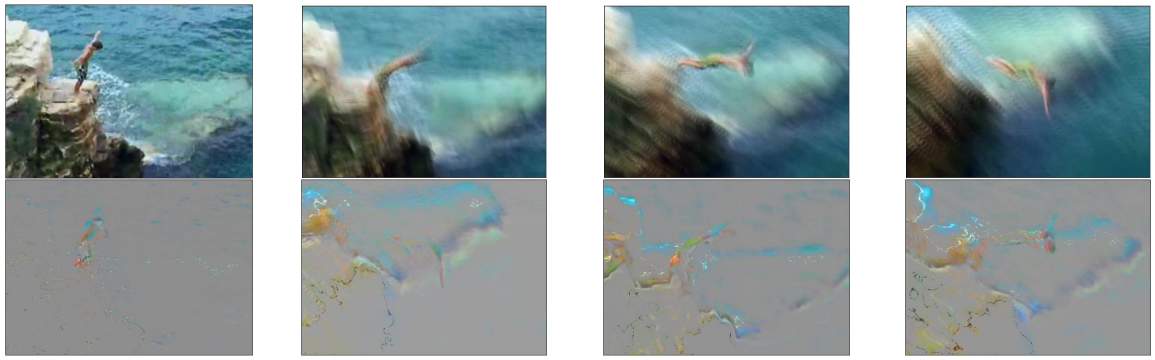
Fig. 3.18 Example videos of 4 segments visualized as their frame averaging and corresponding dynamic appearances.

Tango dancing (Kinetics)

Riding a bike (Kinetics)



Brush hair (HMDB)



Dive (HMDB)

Fig. 3.19 Example videos of 4 segments visualized as their frame averaging and corresponding dynamic appearances.

3.7 Conclusion

In this chapter, we have explored innovative video representation schemes while aiming to improve the models on video understanding.

In the first part of this chapter, we have described a novel video representation, called Squeezed Image, which captures the dynamics of long video sequences useful for video understanding. Meanwhile, we have proposed the Temporal Squeeze Pooling (TSP), which is used to generate the squeezed images. By embedding the TSP module into off-the-shelf CNNs, we proposed the Temporal Squeeze Network (TeSNet) architecture, which learns spatio-temporal features characteristic to discriminating classes of video sequences. We have investigated various locations in the structure of the CNN network for embedding the TSP layers. TeSNet is also used on the optical flow data stream, the prediction score of which is then fused with the RGB image stream to produce higher accuracy. Experiments have been performed on both UCF101 and HMDB51 datasets, and the results indicate that the new video representations are compact and meaningful for improving the action recognition performance. Furthermore, we have analyzed the limitations of the proposed temporal squeeze pooling method. The first disadvantage of TSP is that its output would change the temporal order. Meanwhile, the squeezed images cannot appropriately present motion clues when fast camera movements are involved in videos. Nevertheless, the TSP algorithm is still a preferable solution for fast video recognition applications, as it accelerates the video classification process by reducing the analysis of a video to the analysis of a few images.

In order to expand the application scope of the temporal squeeze pooling method to "motion-dominant datasets", we have upgraded the temporal squeeze pooling method in an innovative way and have proposed the Pixel-Wise Temporal Projection (PWTP). PWTP is a lightweight and backbone-agnostic module, which can achieve the disentanglement of video static and dynamic appearance. The extracted dynamic appearance (DA) characterizes movement by summarizing the visual appearance information that changes from frame to frame. We integrate the PWTP module with a CNN into an efficient and effective spatio-temporal architecture, set in an end-to-end trainable framework. The multiple gradient descent algorithm is used for the joint training of PWTP and the deep network. With extensive experimental results on multiple challenging video benchmarks, we have demonstrated that the proposed dynamic appearances are a qualified input resource to deep networks, showing great advantages over RGB frames and the optical flow inputs, in terms of efficiency and effectiveness. With the ablation studies, we have demonstrated that DA produces higher accuracy than the squeezed images, dynamic images [9] and other popular video representations. The proposed

video representation methodology can contribute to designing optimal spatio-temporal video modeling systems.

Chapter 4

Attention Mechanisms for Long-Range Dependency Modeling in Space and Time

4.1 Introduction

In the analysis of spatial or time series data, long-range dependence can be regarded as a long memory that remembers the information in a long distance or period. Long-range dependency modeling is a common problem in natural language processing (NLP). For instance, to predict the future word in an uncompleted sentence, a network must have a better knowledge about the words prior to it. Long-range dependency modeling methods include recurrent neural networks, Long-Short-Term Memory (LSTM) and attention mechanisms. In video recognition, long-range dependency can also be referred to as global context, as the range covers the full video length and the full spatial sizes. Traditionally, a CNN-based video model captures long-range dependencies by deeply stacking convolutional operations with small window sizes. For example, the convolutional layer with the kernel size of 3×3 has become the standard layers in many current CNN architectures [72, 54, 62]. However, the convolution is a local operation, and the deep stack of local operations limits the efficiency of message delivery to distant positions within a CNN architecture, and makes the optimization difficult [54, 57]. Long-range dependency modeling is more challenging in 3D CNNs than 2D CNNs due to the complexity of input video data across both space and time.

Enabling better global context modeling in video architectures is one of the main contributions of this chapter. An effective way to improve the long-range dependency modeling ability of CNN-based models is to introduce attention mechanisms into the video architectures. Attention mechanisms have been initially used for machine translation [5]. Recent published

works [59, 132, 145, 149] embed task-specific attention mechanisms into CNNs to boost the model performance in visual tasks. In computer vision, attention mechanisms can be decomposed into two components: channel attention - focusing on 'what' is meaningful, and spatial (or spatio-temporal) attention - focusing on 'where' is informative [149]. One representative channel attention mechanism is the Squeeze-and-Excitation (SE) module [59], which utilizes global average-pooled features to exploit the inter-channel relationships. Recently, Wang *et al*. [145] introduced the self-attention concept [131] from machine translation to large-scale visual classification tasks. The non-local (NL) operation was adapted as a spatio-temporal attention mechanism and employed in video recognition [145]. However, Cao *et al*. [15] observe that NL can only capture the global context of channels, aka channel attention, after a series of complex operations. Moreover, they demonstrate that the intrinsic natures of the NL operation and SE module [59] are the same while the implementation of the SE module is rather computationally economical.

In this chapter, we propose two new visual attention mechanisms, namely Convolution Pyramid Attention (CPA) module and Region-based Non-local (RNL) operation, both of which are backbone-agnostic. They are plug-and-play modules that could be embedded into off-the-shelf CNN architectures in order to model long-range dependencies in space-time. The proposed Convolution Pyramid Attention (CPA) adopts the idea from [149] and extends the method from the spatial domain to the spatio-temporal domain. Woo *et al*. [149] only employ a single convolution operation for modeling the spatial attention. In the proposed CPA module, we leverage a pyramid of 3D convolutions with different kernel sizes to capture multi-scale attention in the spatio-temporal dimensions, showing high robustness at various scales. The Region-based Non-local (RNL) operation is an improved version of the Non-local operation [145], which works as a self-attention mechanism [5] that recalibrates the feature at a location according to the information from all other locations in space and time. The proposed RNL operation endues CNNs with a global view of spatio-temporal features, which alleviates the optimization difficulty caused by the deep stack of local operations. Yue *et al*. [155] also aimed to improve the NL operation, proposing a compact generalized version of the NL operation by integrating channel attention and spatio-temporal attention into a compact module. However, their work does not improve the effectiveness of the NL operation. Instead of simplifying the NL, we focus on improving the effectiveness of NL for better capturing the spatio-temporal attention. Through a large number of experiments and quantitative analysis in action recognition tasks, we demonstrate the high efficiency and effectiveness of the proposed CPA and RNL.

The rest of the chapter is organized as follows:

- In Section 4.2, we first revisit the definition of the non-local (NL) operation [145] for long-range dependency modeling. After that, we propose the region-based non-local operation, showing high efficiency and effectiveness in global context modeling.
- In Section 4.3, we describe the proposed Convolution Pyramid Attention (CPA) module that learns vision attention by leveraging the aggregation of filters of various sizes.
- In Section 4.4, we detail the design of a vision attention chain that consist of a spatio-temporal attention module and a channel attention module.
- In Section 4.5 we describe the network architecture configured with our attention modules.
- In Section 4.6, we perform experiments in action recognition tasks on two video benchmarks. We also provide ablation studies for the RNL and CPA modules. A large number of experimental results and visualization analyses are provided to evaluate our methods.
- In Section 4.7, we discuss the shortcomings and limitations of our work.
- In Section 4.8, we draw a short conclusion for this chapter.

4.2 Non-local Methods as Self-Attention

In this section, we introduce region-based non-local (RNL) operations as a family of self-attention mechanisms, which can directly capture long-range dependencies without using a deep stack of local operations. Given an intermediate feature map of a hidden convolutional layer, our method recalibrates the feature at a position by aggregating the information from the neighboring regions of all positions. By combining a channel attention module with the proposed RNL, we design a vision attention chain, which captures the feature attention in the spatio-temporal and channel dimensions.

Wang *et al.* proposed the non-local (NL) operation [145] that works as a self-attention mechanism [131] to capture long-range dependencies directly by exploiting the inner-interactions between different positions regardless of their location difference, which we revisit in Section 4.2.1. However, in the NL operation, the calculation of the relation between two positions

(a) RNL attention maps (b) NL attention maps

Fig. 4.1 Examples of visualizing the attention maps of RNL and NL operations in res4 stage of ResNet on a video clip from Kinetics400. Given a reference position, an ideal non-local operation should only highlight the regions related to that reference position. In the same video clip, the NL operation has almost the same attention maps at different reference positions while the proposed RNL operation presents query-specific attention maps, which demonstrate that the proposed RNL operation is better at capturing positional relationships than the NL operation.

only relies on the information from these two positions while not fully utilizing the information around them. As a result, its calculation of positional relationships is not robust to noise or unrelated features, especially in a high resolution, which has been emphasized in [14]. Here, we investigate the non-local operation [145] and propose a region-based non-local (RNL) operation based on the non-local mean concept [14], which enhances the calculation of positional relationships by fully utilizing the information from neighboring regions. The proposed RNL operation endows CNNs with a global view of input features without requiring a deep stack of local operations. In Fig. 4.1, we illustrate an example to demonstrate that the proposed RNL operation is better at capturing positional relationships than the NL operation.

There are two advantages of the proposed RNL compared with the original NL: first of all, RNL is more robust to noise or unrelated features; secondly, the RNL is more computationally efficient. Meanwhile, we present various instantiations of the RNL operation to meet different application requirements. By embedding RNL modules into the off-the-shelf CNNs, we obtain a new video architecture named the region-based non-local network. In order to evaluate the effectiveness of our method, we conduct action recognition experiments on two large-scale video benchmarks, Kinetics400 [16] and Something-Something V1 [48]. Our models outperform the baseline and other popular attention mechanisms.

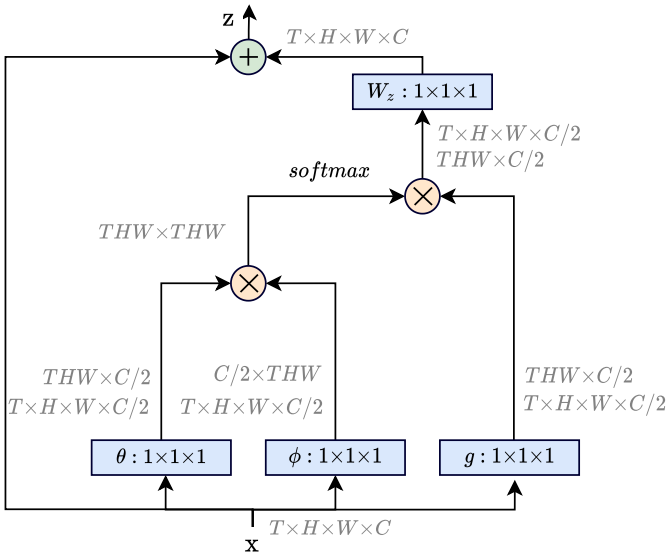


Fig. 4.2 Diagram showing the implementation of the NL operation [145], indicating the shaping and the reshaping operations of a tensor together with the connections. $\otimes$ denotes matrix multiplication while $\oplus$ denotes element-wise addition. The blue boxes denote $1 \times 1 \times 1$ convolutions.

4.2.1 Revisiting the Non-local (NL) Operation

Intuitively, the non-local (NL) operation [145], illustrated in Fig. 4.2 (b), strengthens the feature in a certain position by aggregating the information from all the positions in space or space-time. The estimated value for a position, is computed as a weighted sum of the feature values of all the positions. Formally, we denote $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{THW \times C}$ as the input and output of an NL operation, flattened along the space-time directions, where T, H, W and C are temporal length (depth), height, width and the number of channels, respectively. Then, the NL operation can be described as:

$$\begin{aligned} \mathbf{y}_i &= \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall j} w_{i,j} \mathbf{W}_g \mathbf{x}_j, \\ w_{i,j} &= f(\mathbf{x}_i, \mathbf{x}_j), \end{aligned} \tag{4.1}$$

where $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^C$ are the i-th and j-th element of $\mathbf{x}$, i is the index of a reference position, and j itemizes all possible positions. $\mathbf{W}_g$ is a learnable weight matrix that computes a representation of $\mathbf{x}_j$, and $\mathcal{C}(\mathbf{x})$ is the normalization factor. Meanwhile, $w_{i,j}$ is a weight, representing the relationship between positions i and j, which is calculated by the pairwise

similarity function $f(\cdot,\cdot)$. Regarding the forms of $f(\cdot,\cdot)$ and $\mathcal{C}(\mathbf{x})$, Wang *et al.* [145] propose four instantiations of NL, which are described as follows:

(i) **Gaussian.** $f(\mathbf{x}_i,\mathbf{x}_j) = e^{\mathbf{x}_i^{\mathsf{T}}\mathbf{x}_j}$, $\mathcal{C}(\mathbf{x}) = \Sigma_{\forall j} f(\mathbf{x}_i,\mathbf{x}_j)$;

(ii) **Embedded Gaussian.** $f(\mathbf{x}_i,\mathbf{x}_j) = e^{\theta(\mathbf{x}_i)^{\mathsf{T}}\phi(\mathbf{x}_j)}$, $\mathcal{C}(\mathbf{x}) = \Sigma_{\forall j} f(\mathbf{x}_i,\mathbf{x}_j)$;

(iii) **Dot Product.** $f(\mathbf{x}_i,\mathbf{x}_j) = \theta(\mathbf{x}_i)^{\mathsf{T}}\phi(\mathbf{x}_j)$, $\mathcal{C}(\mathbf{x}) = THW$;

(iv) **Concatenation.** $f(\mathbf{x}_i,\mathbf{x}_j) = \mathrm{ReLU}(\mathbf{w}_f^{\mathsf{T}}[\theta(\mathbf{x}_i),\phi(\mathbf{x}_j)])$, $\mathcal{C}(\mathbf{x}) = THW$.

In the above instantiations of NL, θ, ϕ and $\mathbf{w}_f$ represent linear transformations, which are implemented as 1×1 convolutions in space or $1\times1\times1$ convolutions in space-time. The results from [145] indicate that the performances of the four installations of the non-local operation perform no obvious differences in terms of classification accuracy. In contrast to the original implementation, we would like to emphasize the importance of choosing the right form of function $f(\cdot)$. In our proposed region-based non-local (RNL) operation, described in 4.2.3, we introduce a novel form of function $f(\cdot)$, showing higher performance than the previous four instantiations of NL provided above.

4.2.2 Attention Maps of the Non-local (NL) Operation

Within the NL operation [145], each output element $\mathbf{y}_i$ is a weighted average of the input features over all positions $\mathbf{x}_j$, and therefore each $\mathbf{y}_i$ has a corresponding attention weight map calculated by $f(\cdot,\cdot)$, highlighting the areas related to position i. In Fig. 4.1 (b), we randomly pick one video from Kinetics400 and visualize the attention maps of NL at two different reference positions, one of which is located in the background area while the other is located in the region of the moving object. From Fig. 4.1 (b), we can observe that the attention maps of NL, with respect to different reference locations, fail to capture the relationships between the information from different locations. We redesign the non-local operation as a spatio-temporal attention mechanism, namely the region-based non-local operation (RNL). Fig. 4.1 (a) shows that our RNL operation highlights only the regions related to the reference position, which indicates that the proposed RNL operation can effectively model the positional relationships in the spatio-temporal domain.

4.2.3 Region-based non-local (RNL) Operation

The initial idea for the RNL operation is that the relation between two positions in a video representation should not rely on just their own features but also on the features from their

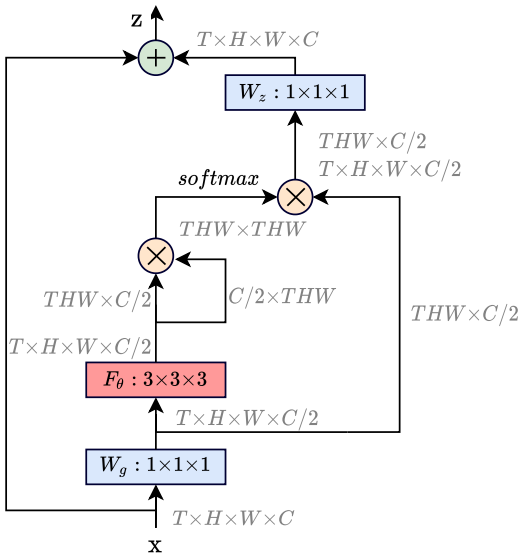


Fig. 4.3 Diagram showing the implementation of the proposed RNL operations, indicating the shaping and the reshaping operations of a tensor together with the connections. $\otimes$ denotes matrix multiplication while $\oplus$ denotes element-wise addition. The blue boxes denote $1 \times 1 \times 1$ convolutions, and the red box F_θ denotes a $3 \times 3 \times 3$ channel-wise convolution or an average/max pooling layer.

neighborhoods. Therefore, for each position i of input sample $\mathbf{x}$, we define a cuboid region $\mathcal{N}_i$ of fixed size centered at position i. The calculation of the relationship $w_{i,j}$ between positions i and j is redefined as:

$$w_{i,j} = f(\theta(\mathcal{N}_i), \theta(\mathcal{N}_j)), \tag{4.2}$$

where, $\theta(\cdot)$ denotes an information aggregation function that separately summarizes the features in a region for each channel. Function $\theta(\cdot)$ is defined by

$$\theta(\mathcal{N}_i) = \sum_{k \in \mathcal{N}_i} \mathbf{u}_k \odot \mathbf{x}_k, \tag{4.3}$$

where $\odot$ denotes element-wise multiplication and $\mathbf{u}_k$ denotes a vector shared by all cuboid regions $\mathcal{N}_i$. As there is no channel interaction in $\theta(\cdot)$, it can be implemented as channel-wise convolutions [106], or as average/max pooling. Channel-wise convolution is also referred to as “depth-wise". We use the term “channel-wise" to avoid confusions with the network depth. By replacing the expression of $w_{i,j}$ from Eq. (4.1) with the expression from Eq. (4.2),

the RNL operation can be written as:

$$\mathbf{y}_i = \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall j} f(\theta(\mathcal{N}_i), \theta(\mathcal{N}_j)) \mathbf{x}_j. \tag{4.4}$$

From Eq. 4.4, we can see that by employing the RNL operation, the new feature characteristic for each position is a weighted sum of the old features from all positions, where the weights are calculated by the similarity function $f(\cdot,\cdot)$ according to the similarity between the target region, and all other regions. The proposed RNL operation enhances the calculation of positional relations by fully utilizing the information from the neighboring regions, which increases the robustness to noise or unrelated features. Hence, the RNL operation can learn more meaningful representations in comparison with the original NL operation [145].

For the form of function $f(\cdot,\cdot)$, besides adopting the Gaussian version and the Dot product version as in [145], we also propose a new form, called the Cosine version. Specifically, the **Gaussian** form of $f(\cdot,\cdot)$ is given by

$$f(\theta(\mathcal{N}_i), \theta(\mathcal{N}_j)) = e^{\theta(\mathcal{N}_i)^\intercal \theta(\mathcal{N}_j)}. \tag{4.5}$$

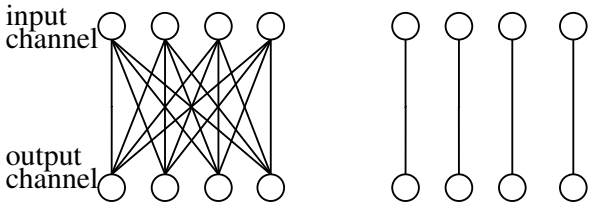
The **Dot product** form of $f(\cdot,\cdot)$ measures the relation between two regions by using the dot-product similarity:

$$f(\theta(\mathcal{N}_i), \theta(\mathcal{N}_j)) = \theta(\mathcal{N}_i)^\intercal \theta(\mathcal{N}_j). \tag{4.6}$$

However, the dot-product similarity takes into account both the vector angle and the magnitude, as $\theta(\mathcal{N}_i)^\intercal \theta(\mathcal{N}_j) = \|\theta(\mathcal{N}_i)\| \|\theta(\mathcal{N}_j)\| \cos \psi_{i,j}$, where $\psi_{i,j}$ is the angle between vectors $\theta(\mathcal{N}_i)$ and $\theta(\mathcal{N}_j)$. It is preferable to replace dot-product similarity with the cosine similarity, ignoring the vector magnitude and resulting in a value within the range $[-1, 1]$. The **Cosine** form of $f(\cdot,\cdot)$ is expressed as:

$$\begin{aligned} f(\theta(\mathcal{N}_i), \theta(\mathcal{N}_j)) &= \mathrm{ReLU}\left(\frac{\theta(\mathcal{N}_i)^\intercal \theta(\mathcal{N}_j)}{\|\theta(\mathcal{N}_i)\| \|\theta(\mathcal{N}_j)\|}\right) \\ &= \mathrm{ReLU}(\cos \psi_{i,j}). \end{aligned} \tag{4.7}$$

When $f(\theta(\mathcal{N}_i), \theta(\mathcal{N}_j)) < 0$, it indicates that the features in positions i and j are not related. As the new feature in a certain position should only be determined by those related features, we use the ReLU function to restrict the output of $f(\cdot,\cdot)$ to be non-negative. The normalization factor is set as $\mathcal{C}(\mathbf{x}) = \sum_{\forall j} f(\theta(\mathcal{N}_i), \theta(\mathcal{N}_j))$ for the **Gaussian** version from Eq. (4.5),



(a) conventional convolution (b) channel-wise convolution

Fig. 4.4 Illustrations of the conventional convolution (a) and the channel-wise convolution (b). The total number of connections of the channel-wise convolution [106] is reduced to $\frac{1}{C}$ of that of the conventional convolution.

and set as $\mathcal{C}(\mathbf{x}) = THW$ for the Dot-product and Cosine versions from Eq. (4.6) and 4.7, respectively.

4.2.4 Region-based non-local Block

In order to embed the RNL operation into the off-the-shelf CNNs without influencing the results provided by the pre-trained kernels, we embed the RNL operation into a residual style block [54], named the RNL block. The Gaussian RNL block, defined by Eq. (4.5), is written as a matrix form as:

$$\mathbf{z} = \mathbf{y}\mathbf{W}_z + \mathbf{x}, \tag{4.8}$$

$$\mathbf{y} = Softmax(F_\theta(\mathbf{x}\mathbf{W}_g)(F_\theta(\mathbf{x}\mathbf{W}_g))^\intercal)\mathbf{x}\mathbf{W}_g, \tag{4.9}$$

where $\mathbf{z}$ is the output representing the feature after recalibration, $\mathbf{W}_z \in \mathbb{R}^{\frac{C}{2}\times C}$ and $\mathbf{W}_g \in \mathbb{R}^{C\times\frac{C}{2}}$ are learnable weight matrices, which are implemented as $1\times1\times1$ convolutions, and '$+\mathbf{x}$' denotes a residual term. F_θ denotes the operation that corresponds to the matrix form of function $\theta(\cdot)$ from Eq. (4.3). We present the architectures of the Gaussian RNL block and the Gaussian embedding version of the original NL block in Fig. 4.3 and Fig. 4.2, respectively. We can observe that the original NL block uses four $1\times1\times1$ convolutions, while the proposed RNL block shown uses only two $1\times1\times1$ convolutions and one channel-wise convolution, which reduces the computational complexity significantly.

In the following, we explain two implementations for the region information aggregation function F_θ in RNL.

1) Channel-wise Convolutions. It is worthwhile to note that, in principle, the candidates for implementing F_θ should not fuse together information across channels. Otherwise, the new feature embedding might fail to represent the information from each original channel,

which is why we do not adopt conventional convolutions. In contrast, channel-wise convolution [106], exemplified in Fig. 4.4, is a perfect candidate for the implementation of F_θ, as there is no cross-interaction between the channels. An additional benefit that the channel-wise convolution brings is that it reduces the required parameters and computation by a factor of C, compared with the conventional convolution. The kernel size of the channel-wise convolution has a significant impact on performance, as it corresponds to how large a region $\mathcal{N}_i$ is considered for information aggregation. We will explore the effectiveness of various kernel sizes, in Section 4.6.1.

2) Average/Max Pooling. The other implementation options for F_θ are the average pooling and max pooling, which have been widely adopted for information aggregation. Although it shows a relatively weaker capability than the implementation of channel-wise convolution, average/max pooling adds no extra parameters to the models.

4.3 Convolution Pyramid Attention (CPA)

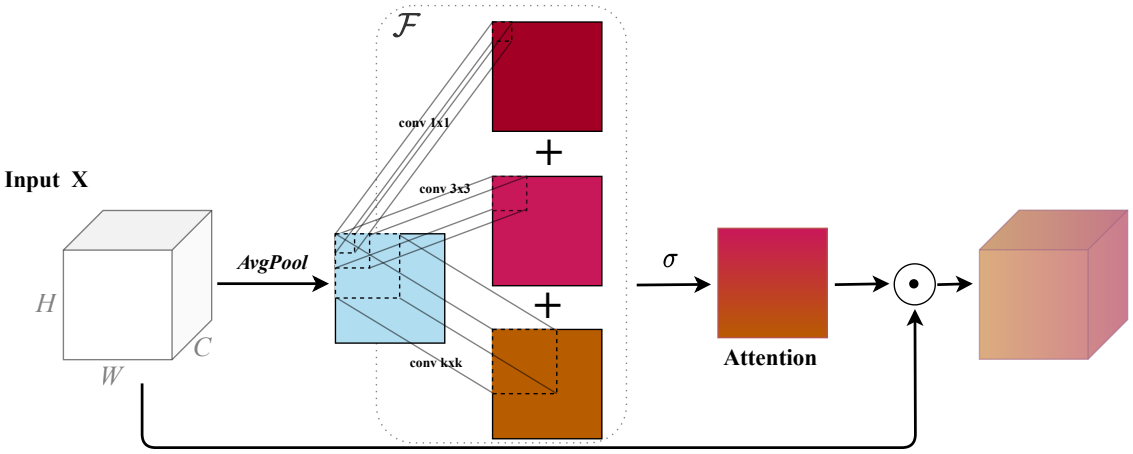


Fig. 4.5 Diagram showing the Convolution Pyramid Attention (CPA) Module. The temporal dimension is omitted in the diagram for simplification. $\mathcal{F}$ denotes a pyramid of convolutions from Eq. (4.11). σ denotes the Softmax operation. $\odot$ denotes element-wise multiplication.

Spatio-temporal attention tells 'where' and 'when' to focus. By introducing attention mechanisms into CNNs, the representations of important features would be enhanced while the tedious ones would be suppressed. As stated in Section 4.2, a shallow convolutional layer with a small kernel working as a local operation would only receive very limited information

within the small window and so would be an obstacle to modeling global context. To address this issue, one intuitive solution is to use the convolution with an unusually large kernel to receive the information from a wider view at the same level of depth for a CNN. However, larger kernels with hundreds of feature channels in a CNN are not an implementation option considering the optimization of the computational resources. We design a vision attention mechanism, which leverages a single-channel convolution with an unusually large kernel to recalibrate the feature values learned previously by the small kernels. We term the attention mechanism Convolution Pyramid Attention (CPA) module. The diagram of CPA is shown in Fig. 4.5. Formally, by considering an input feature map $\mathbf{x}$, the output of CPA module is given by

$$\text{CPA}(\mathbf{x}) = \sigma(\mathcal{F}(AvgPool(\mathbf{x}))) \odot \mathbf{x}, \tag{4.10}$$

where the average pooling operates along the channel axis, $\sigma(\cdot)$ denotes the Softmax operation along the space and time, $\bigodot$ denotes element-wise multiplication. By considering convolutions with different kernel sizes we can learn features at different scales. We define the function $\mathcal{F}(\cdot)$ as a pyramid of convolutions, which is composed of multiple convolutions of different kernel sizes:

$$\mathcal{F}(\mathbf{x}) = f^{1\times1\times1}(\mathbf{x}) + f^{3\times3\times3}(\mathbf{x}) + \ldots + f^{k_t\times k_s\times k_s}(\mathbf{x}), \tag{4.11}$$

where $f^{k_t\times k_s\times k_s}$ refers to a convolution with the kernel of size $k_t \times k_s \times k_s$.

The Inception module [120] also employs filters of multiple sizes. Nevertheless, the proposed CPA is different from the Inception module in both conceptional and functional aspects. The Inception module [120] aims to approximate an optimal local sparse structure in a CNN by concatenating the outputs of multiple convolutions with different kernel sizes forming the input to the next stage. However, the purpose of the convolution pyramid is to learn spatio-temporal attention: the aggregated vector is used to recalibrate the feature values learned by the previous convolutional layers. Meanwhile, the proposed CPA module is not a simple 3D extension of the Convolutional Block Attention Module (CBAM) because we introduce the concept of multi-scale feature learning to the design of CPA. The proposed CPA module is shown through experiments to be more efficient and have better performance than the CBAM. The results reaffirm the significance of the convolution pyramid design in CPA.

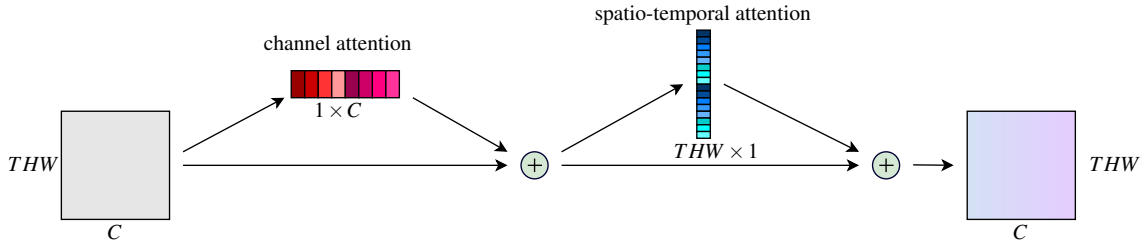


Fig. 4.6 The diagram of the vision attention chain. The channel attention mechanism is implemented as the squeeze-excitation block [59] while the spatio-temporal attention mechanism is implemented as the RNL or CPA block.

4.4 The Vision Attention Chain

When the proposed RNL and CPA modules can learn the long-range dependencies for each position in the spatio-temporal dimensions, the squeeze-excitation (SE) block [59] would learn the long-range dependencies along the channel dimension. In order to capture both spatio-temporal attention and channel-wise attention in a single module, we connect the RNL or CPA blocks together with the SE block [59] to form a vision attention chain, whose diagram is shown in Fig. 4.6. Firstly, we modify the SE block and adapt it [59] to the spatio-temporal domain, where the squeeze operation F_{sq} is expressed as:

$$\mathbf{s}' = F_{sq}(\mathbf{x}) = \frac{1}{THW}\sum_{i=1}^{THW}\mathbf{x}_i, \tag{4.12}$$

while the excitation operation F_{ex} is expressed as:

$$\mathbf{s} = F_{ex}(\mathbf{s}') = \mathbf{W}_2\text{ReLU}(\text{BN}(\mathbf{W}_1\mathbf{s}')), \tag{4.13}$$

where $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{2}\times C}$ and $\mathbf{W}_2 \in \mathbb{R}^{C\times\frac{C}{2}}$ are trainable weights, which can be implemented with fully-connected (FC) layers. In the excitation operation F_{ex}, we add a batch normalization (BN) layer [65] right after the FC layer $\mathbf{W}_1$ to reduce the internal covariate shift. The output of the SE block is given by:

$$\mathbf{v} = \mathbf{x} \oplus \mathbf{s}, \tag{4.14}$$

where $\oplus$ refers to the element-wise addition broadcasting in unmatched dimensions (replicate $\mathbf{x}$ to match the dimension of $\mathbf{s}$). After that, we place the RNL or CPA block after the SE block to form a vision attention chain. The vision attention chain endues standard CNNs with channel attention and spatio-temporal attention, capturing "what", " where" and "when" is informative and meaningful.

4.5 The Network Architecture

Table 4.1 The architecture of the ResNet50 empowered by the RNL and CPA modules. The kernel size and the output size are shown in the second and third columns, respectively. The RNL or CPA blocks are inserted after the res3 and res4 blocks, while the temporal shift modules [84] is embedded into the first convolutional layer in each residual block.

Layer	Operation	Output size
conv1	$1 \times 7 \times 7$, 64, stride 1,2,2	$8 \times 112 \times 112$
pool1	$1 \times 3 \times 3$, 64, stride 1,2,2	$8 \times 56 \times 56$
res2	$\begin{bmatrix} 1 \times 1 \times 1, 64 \\ 1 \times 3 \times 3, 64 \\ 1 \times 1 \times 1, 256 \end{bmatrix} \times 3$	$8 \times 56 \times 56$
res3	$\begin{bmatrix} 1 \times 1 \times 1, 128 \\ 1 \times 3 \times 3, 128 \\ 1 \times 1 \times 1, 512 \end{bmatrix} \times 4$ RNL / CPA	$8 \times 28 \times 28$
res4	$\begin{bmatrix} 1 \times 1 \times 1, 256 \\ 1 \times 3 \times 3, 256 \\ 1 \times 1 \times 1, 1024 \end{bmatrix} \times 6$ RNL / CPA	$8 \times 14 \times 14$
res5	$\begin{bmatrix} 1 \times 1 \times 1, 512 \\ 1 \times 3 \times 3, 512 \\ 1 \times 1 \times 1, 2048 \end{bmatrix} \times 3$	$8 \times 7 \times 7$

The RNL and CPA blocks are designed to be compatible to be used with most existing CNNs. It can be easily plugged into off-the-shelf CNNs at any processing stage, resulting in more powerful network architectures with stronger global context modeling abilities. For the implementation, we use ResNet50 [54] with the temporal shift modules (TSM) [84] as the backbone network to build our models (RNL TSM and CPA TSM). The resulting network structures are outlined in Table 4.1. The TSM is a lightweight module enabling 2D CNNs to achieve temporal modeling by shifting part of the channels along the temporal dimension, which facilitates the information exchange among neighboring frames. In this architecture, we keep the temporal size constant, which means that all the layers in the network only reduce the spatial size of the input features. The backbone network (TSM ResNet50) is used as the baseline for our experiments.

4.6 Experiments

In the following we provide the experimental results for the proposed attention-based model for action recognition in videos. We perform action recognition experiments on two standard

video benchmarks, Kinetics400 [16] and Something-Something V1 [48]. We report Top-1, Top-5 accuracy on the validation sets and the computational cost (in GFLOPs) of a single, spatially center-cropped clip to comprehensively evaluate the effectiveness and efficiency. The experimental results of our method outperform other attention mechanisms. The implementation code for this chapter is available at: *https://github.com/guoxih/region-based-non-local-network*.

Training and Inference

Our models are pretrained on ImageNet [22]. For the training, we follow the training regime from [145] and use a spatial size of 224 $\times$ 224. The temporal size is set as 8 frames unless otherwise specified. In order to prevent overfitting, we add a dropout layer after the global pooling layer. We optimize our models using the Stochastic Gradient Descent, and train the models for 50 epochs with a cosine decay learning rate schedule. The batch size is set at 64 across multiple GPUs. For Kinetics, the initial learning rate, weight decay and dropout rate are set to 0.01, 1e-4 and 0.5 respectively; for Something-Something, these hyper-parameters are set to 0.02, 8e-4, and 0.8 respectively. The training process of RNL network is shown in Fig. 4.7, where the curves show the Softmax cross-entropy loss at the clip-level prediction. For the inference, we follow the common setting in [145, 84]. Unless stated otherwise,

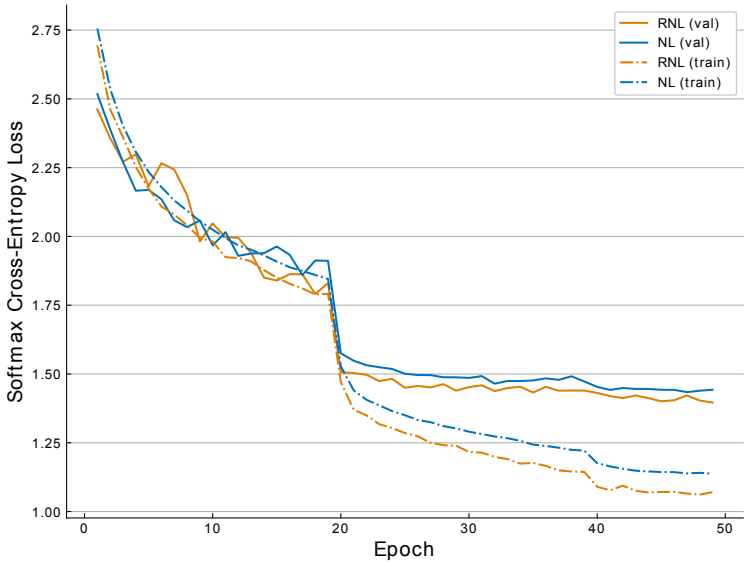


Fig. 4.7 Training the network with 5 NL blocks and with 5 RNL blocks, respectively, on Kinetics400. Our RNL network shows improved optimization characteristics throughout the training process. We calculate the Softmax cross-entropy loss on both training set (train) and validation set (val).

we uniformly sample 10/2 clips for Kinetics400/Something-Something V1, and perform spatially fully convolutional inference (three crops of size 256×256 to cover the spatial dimensions) for all clips, while the video-level prediction is obtained by averaging all the clip prediction scores of a video.

4.6.1 Ablation Studies for RNL

We aim to find out the most efficient and effective form of RNL. By default, the function $f(\cdot,\cdot)$ adopt the Gaussian form in Eq. (4.5), and F_θ is implemented by a channel-wise convolution with a kernel size of $3 \times 7 \times 7$, unless otherwise specified. Following the results from [145], we add RNL blocks to the res3 and res4 stages in the architecture shown in Table 4.1. Our exploration is organized in three parts. First, we search for the effective kernel size of F_θ in RNL blocks. Next, we evaluate the performance of various instantiations of RNL and find out the most efficient and effective one. Finally, we combine the selected version of RNL with an SE block to form a vision attention chain. The results are provided on the Kinetics400 dataset.

Kernel Size of RNL

We explore the influences of various kernel sizes of RNL on the performance. Large kernels are supposed to be robust to noise, while small kernels would consider the details and fine structures from video sequences but may be too focused. The results are shown in Table 4.2. We observe that the kernel of $3 \times 7 \times 7$ provides the best result (73.66%) among the listed settings. Concurrently, we evaluate the influence of the kernel size of F_θ on the model

Table 4.2 Ablation for the RNL operations with various kernel sizes of F_θ, which is implemented as a channel-wise convolution operation. We insert one Gaussian RNL block into the res3 stage of ResNet50.

Kernel size	Top-1 (%)	Kernel size	Top-1 (%)
$1 \times 1 \times 1$	73.28	$3 \times 3 \times 3$	73.53
$3 \times 1 \times 1$	73.41	$3 \times 5 \times 5$	73.27
$7 \times 1 \times 1$	73.12	$\mathbf{3 \times 7 \times 7}$	**73.66**
$1 \times 3 \times 3$	73.32	$3 \times 9 \times 9$	73.51
$1 \times 7 \times 7$	73.43	$7 \times 7 \times 7$	73.11
$1 \times 9 \times 9$	73.32	$7 \times 9 \times 9$	73.30

performance by visualizing the attention maps of the RNL operation, shown in Fig. 4.8. The RNL operation considers the highlighted areas to have strong relations with the reference

1×1 3×3 5×5 7×7 9×9

Fig. 4.8 Attention maps of the RNL block when considering different kernel sizes in the res3 stage when providing the reference point, shown as a red point. When the reference point is located at the moving object, the RNL operation with proper kernel size should just highlight the related moving regions.

position, indicated by a red point. Fig. 4.8 shows that a kernel of a small size spatially, such as 1×1, tends to incorrectly interpret the relations between some background areas and the foreground areas. In contrast, a kernel with a larger spatial size can learn more precise relations between such positions. For example, the kernel of size 7×7 precisely highlights the moving object in in Fig. 4.8 when the reference position is located at the moving object.

Instantiations of RNL

Table 4.3 Instantiations of RNL with different implementations of F_θ. We insert one Gaussian RNL block into the res3 stage of ResNet50.

Method (F_θ)	Top-1 (%)	GFLOPs	Params
channel-wise conv	**73.66**	1.65	2.67M
average pooling	73.22	1.65	0.26M
max pooling	73.47	1.65	0.26M

There are various solutions for $f(\cdot,\cdot)$ from Eq. (4.4) and for F_θ from Eq. (4.9), as discussed in Section 4.2.3 and Section 4.2.4, respectively. In the following, we conduct ablation studies on the instantiations by fixing a specific choice for either $f(\cdot,\cdot)$ or F_θ while changing the other. The operation F_θ can be implemented as a channel-wise convolution or as the average/max pooling, the stride of which is set as 1, and the padding of which is half of the kernel size. From the results shown in Table 4.3, we can see that the channel-wise convolution implementation achieves a higher accuracy with +0.44% and +0.19% than the average and max pooling, respectively. However, the implementation of average/max pooling is more efficient and adds fewer parameters (-2.4M) to the model compared to the channel-wise convolution. We instantiate three versions of the RNL operation, such as Gaussian, Dot-product and Cosine, provided in Eq. (4.5), (4.6) and (4.7) respectively. The results are

shown in Table 4.4. By adding a single RNL block into the backbone network, the Cosine

Table 4.4 Instantiations of the RNL with different form of $f(\cdot,\cdot)$.

# RNL	Method($f(\cdot,\cdot)$)	Top-1 (%)
1	Dot-product	73.22
	Gaussian	**73.66**
	Cosine	73.46
5	dot-product	74.16
	Gaussian	**74.68**
	Cosine	74.40

RNL operation produces higher accuracy than the Dot-product RNL operation due to the additional term of normalization from Eq. (4.7). The Gaussian RNL achieves the highest accuracy among the RNL variants. Moreover, the performance of all installations of RNL can be further boosted by stacking more RNL blocks. The model with 5 Gaussian RNL blocks (3 in the res4 stage and 2 in the res3 stage) gains an additional 1.02% accuracy increase in comparison with when adding a single RNL block.

Visualization

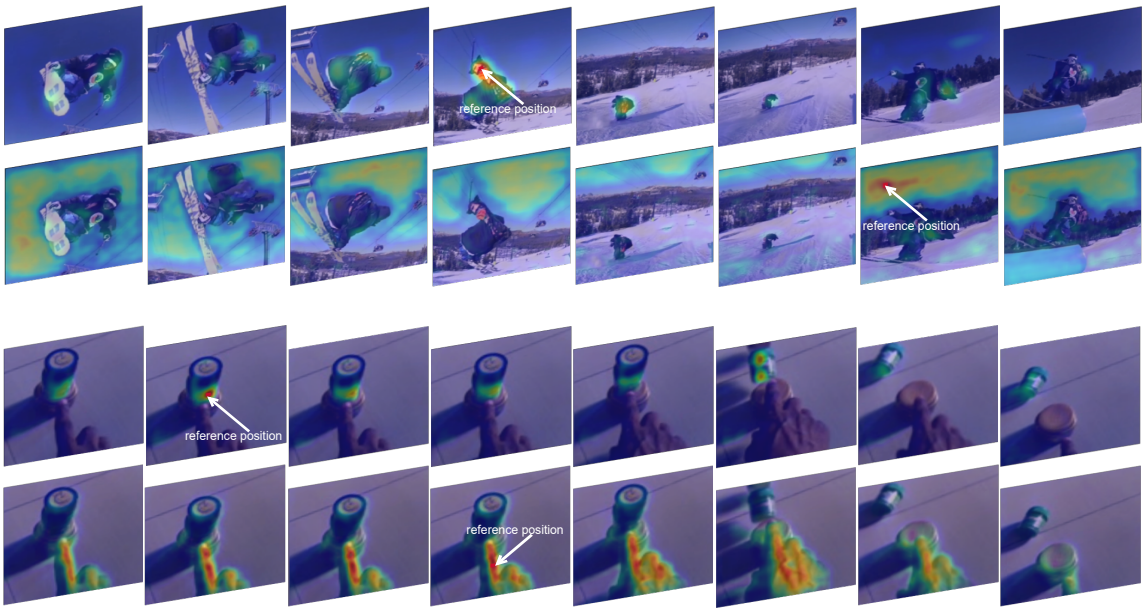


Fig. 4.9 Example attention maps of RNL in the res3 stage, with different reference positions on frames from Kinetics (1st row) and Something-Something (2nd row). Given a video clip, the RNL operation only highlights those regions related to the reference position.

In Fig. 4.9, we visualize some example attention maps of RNL in the res3 stage, which shows that the RNL operation can correctly learn the relationships between different positions in videos. In the example shown at the bottom, where a hand touches a battery placed upright on a surface, we can observe that the touched region of the battery is highlighted when the reference position is also located on the region of the battery from the image. Similarly, when the reference position is located in the fingers, then the region of the human hand is highlighted in the image. In addition, Fig. 4.10 visualizes more example attention maps of RNL in the res4 stage. We observe that the RNL operation cannot only precisely learn the feature relationships between spatio-temporal positions in the res3 stage but also when inserted in the res4 stage. This observation suggests that the RNL operations that are embedded at different depths of the backbone network perform the same function resulting in an excellent global context modeling ability.

4.6.2 Ablation Studies for CPA

In the CPA module, the convolution pyramid described in Sec. 4.3, which is implemented with $\mathcal{F}(\cdot)$ from Eq.(4.11), is vital for the enhanced ability to model the global context in space-time. The convolution pyramid, consisting of multiple convolution filters with different kernel sizes, enables the CPA module to effectively model long-range dependencies in a multi-scale manner. Here, we ablate the convolution pyramid in the CPA module, by considering kernels of different size and combinations of various filters. The results are provided in Table 4.5. For the models in Table 4.5, we add 5 CPA blocks (3 blocks to res4 stage and 2 blocks to res3 stage) to the backbone network same as in the ablation experiments for RNL in Sec. 4.6.1. A larger kernel $\mathcal{F}$ yields a higher performance gain, according to the results from the top side of Table 4.5. Consequently, larger kernels can enable better global context modeling. Moreover, we build a hybrid filter $\mathcal{F}$ by aggregating multiple single-channel convolutions of different kernel sizes, the results of which are shown in the bottom part of Table 4.5. We note that all the listed convolution pyramid designs except $\mathcal{F} = f^{3\times3\times3} + f^{3\times7\times7}$ work better than the instantiation of any single kernel, which emphasizes the importance of using the convolution pyramid design in CPA.

Although the CPA instantiation with $\mathcal{F} = 1 \times 1 \times 1$ does not improve performance, the integration of a kernel of size $1 \times 1 \times 1$ with other larger kernels results in a strong multi-scale attention learning behavior. For example, the CPA instantiation with $\mathcal{F} = f^{1\times1\times1} + f^{3\times3\times3} + f^{3\times7\times7}$ generates higher accuracy than $\mathcal{F} = f^{3\times3\times3} + f^{3\times7\times7}$.

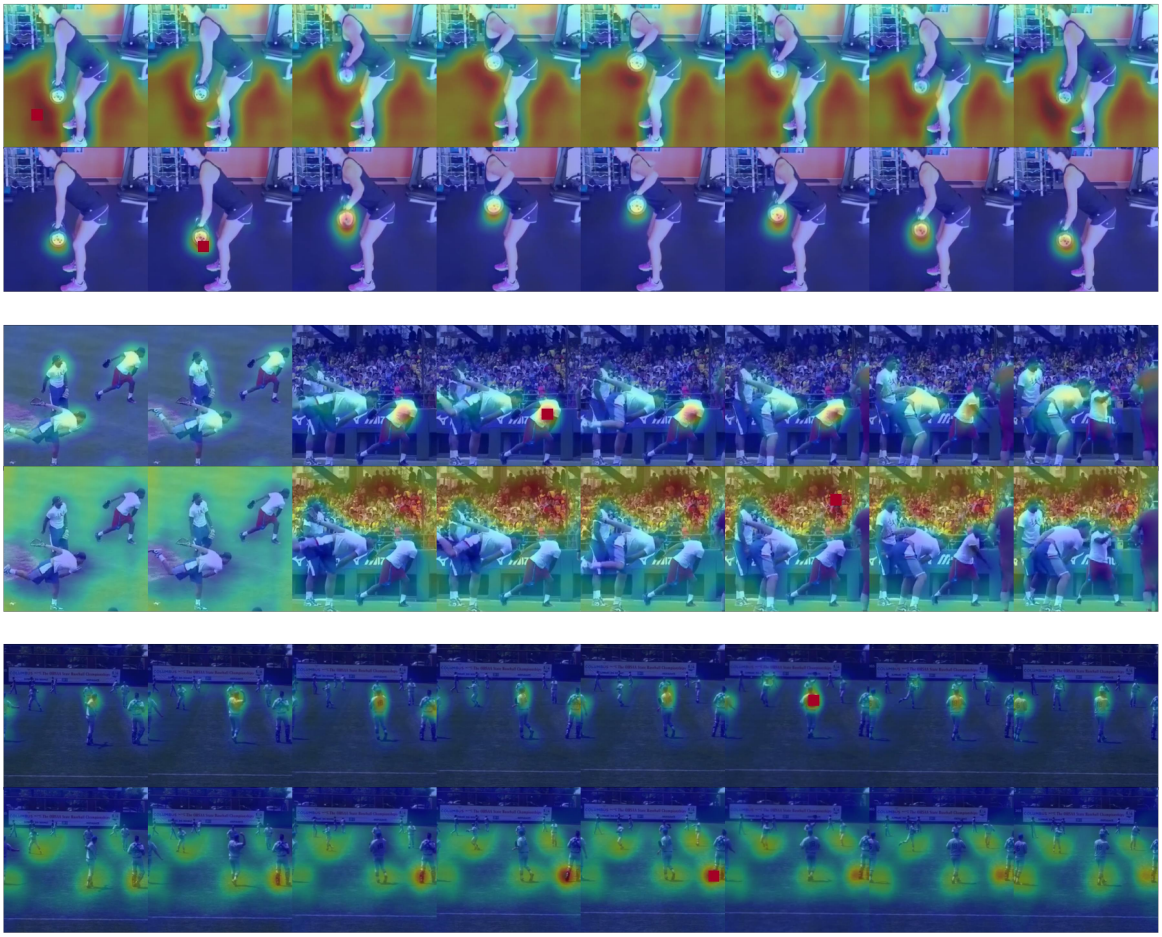


Fig. 4.10 Example attention maps of the RNL in the res4 stage, with different reference positions (red points) on frames from Kinetics.

Visualization

For the attention map visualization, we consider applying the Gradient-weighted Class Activation Mapping (Grad-CAM) [109] to the networks. Grad-CAM utilizes the gradients of a particular convolutional layer with respect to a given target to produce coarse localization maps, highlighting important regions. The visualization can be used to identify failing models. We compare the visualization results of the CPA-integrated network (CPA+ResNet50) with that of the baseline (TSM ResNet50) to further evaluate our method. Some visualization examples are shown on Fig. 4.11. We can observe that the CPA-integrated network can indicate better the regions where the activities take place than the network without CPA.

Table 4.5 Ablation for the convolution pyramid ($\mathcal{F}$) on Kinetics400. We embed 2 CPA blocks into res3 stage and 3 CPA blocks into res4 stage of ResNet50.

		Accuracy (%)
w/o CPA	-	72.80
w/ CPA	$\mathcal{F} = f^{1\times1\times1}$	72.53
w/ CPA	$\mathcal{F} = f^{3\times3\times3}$	73.52
w/ CPA	$\mathcal{F} = f^{3\times7\times7}$	73.87
w/ CPA	$\mathcal{F} = f^{3\times9\times9}$	73.92
w/ CPA	$\mathcal{F} = f^{3\times3\times3} + f^{3\times7\times7}$	73.86
w/ CPA	$\mathcal{F} = f^{1\times1\times1} + f^{3\times3\times3} + f^{3\times7\times7}$	<u>74.01</u>
w/ CPA	$\mathcal{F} = f^{1\times1\times1} + f^{3\times3\times3} + f^{3\times7\times7} + f^{3\times9\times9}$	**74.27**

Moreover, the prediction scores of the target action classes increase accordingly after being configured with CPA modules.

4.6.3 Evaluation

In order to evaluate the efficiency and effectiveness of our methods in comparison with other attention mechanisms, we reimplement the original NL network [145], GCNet [15] (a simplified NL network), SE network [59] and CBAM network [149]. Table 4.6 presents the

Table 4.6 Comparisons between various visual attention mechanisms on Kinetics400 and Something-Something V1.

Dataset	Model	Top-1 (%)	FLOPs (G)	# Param (M)
Kinetics-400	baseline	72.80	32.89	24.33
	+ 5 SE	73.70	32.89	24.79
	+ 5 CBAM	73.99	32.90	24.80
	+ 5 GC	73.76	32.90	24.79
	+ 5 NL	74.41	49.38	31.69
	+ 5 CPA	74.27	-	-
	+ 5 [SE+CPA]	74.33	-	-
	+ 5 RNL	74.68	41.15	35.48
	+ 5 [SE+RNL]	**74.97**	41.16	35.95
Something-Something V1	baseline	46.63	32.89	24.33
	+ 5 NL	48.25	49.38	31.69
	+ 5 RNL	49.24	41.15	35.48
	+ 5 [SE+RNL]	**49.47**	41.16	35.95

results on the Kinetics and Something-Something datasets. We can see that the proposed

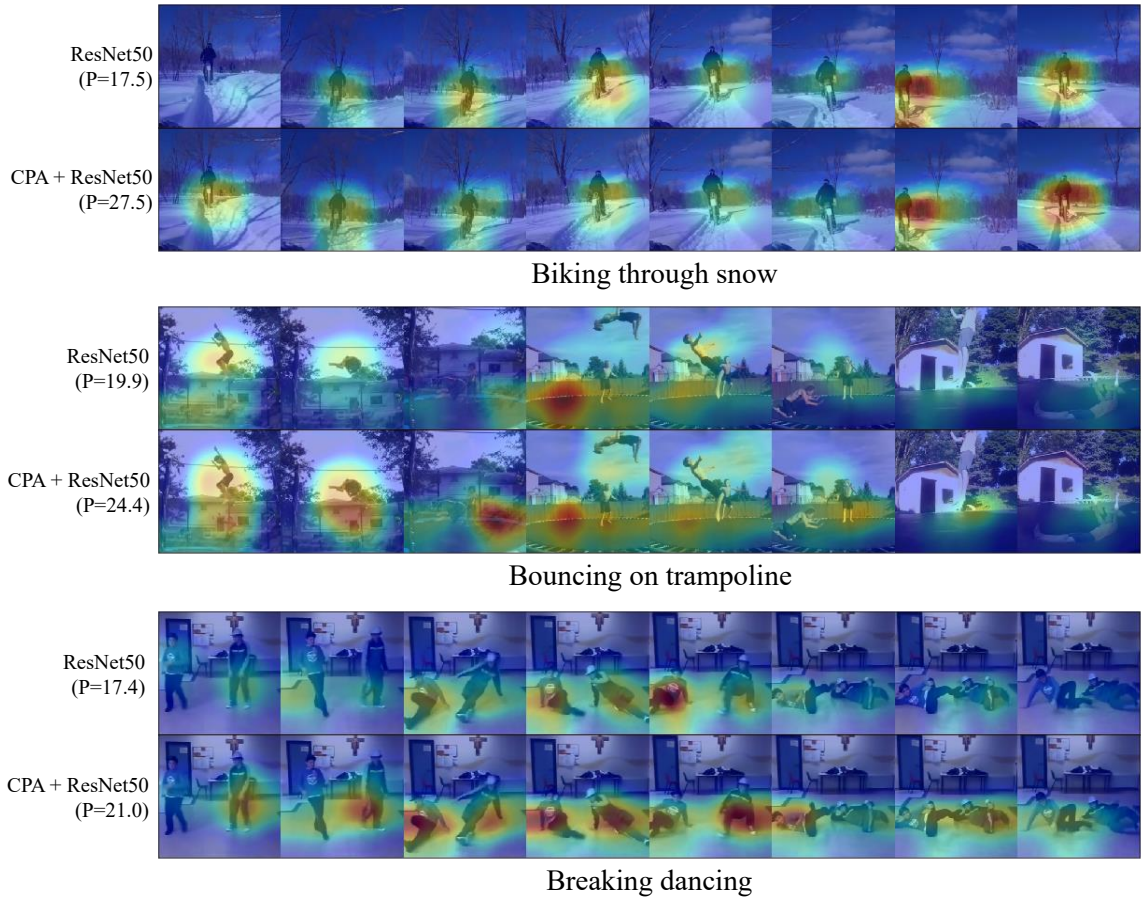


Fig. 4.11 Grad-CAM visualization examples of CPA-integrated network (CPA+ResNet50) and ResNet50 with TSM [84]. The heatmaps highlight the salient regions. The Grad-CAM masks are calculated on the last convolutional layers. The ground-truth label is shown on the bottom of each pair of Grad-CAM masks. The prediction score of the target class is shown within the brackets.

RNL block achieves higher performance than other attention mechanisms. The network with 5 RNL blocks outperforms the network with 5 NL blocks with +0.27% on Kinetics and +1% on Something-Something, while the required computational complexity of the RNL network is 8.23GFLOPs less than that of the NL network. Furthermore, by adding 5 blocks of the vision attention chain (SE + RNL), as described in Section 4.4, to the backbone network, the performance is further improved (74.97% on Kinetics and 49.47% on Something-Something). In the visualization examples of the RNL and NL blocks, shown in Fig. 4.1, we observe that the attention maps of the RNL block would only highlight those regions related to the reference positions. However, the attention maps of the original NL block always highlight the same regions for different reference positions. This observation demonstrates that the

RNL block can better capture the spatio-temporal attention than the NL block. According to the results on Kinetics400, the model that has CPA or CBAM module embedded achieves lower performance than the self-attention mechanisms NL [145] and RNL. This suggests that self-attention mechanisms are a type of more efficient approach for modeling long-range dependencies. Nevertheless, the CPA-integrated network still achieves higher performance than SE [59] and CBAM [149].

4.6.4 Comparisons

Considering that the proposed RNL operation performs better than the CPA module, we choose RNL as our representative and compare it with the state-of-the-art methods on Kinetics400 and Something-Something V1. In order to achieve the best performance on Kinetics400, we increase the number of training epochs from 50 to 100. The performance comparisons are summarized in Tables 4.7 and 4.8, where RNL TSM refers to the model with 5 vision attention chain blocks. Note that using the same approach, the models with

Table 4.7 Comparisons with SOTA on Kinetics400.

Model	Backbone	Training Frames	Top-1 (%)	Top-5 (%)
I3D RGB [16]	Inception	64	72.1	90.3
I3D Flow [16]		64	65.3	86.2
I3D RGB+Flow [16]		64	75.7	92.0
S3D-G RGB [151]		64	74.7	93.4
S3D-G Flow [151]		64	68.0	87.6
S3D-G RGB+Flow [151]		64	77.2	93.0
TSM [84]	ResNet50	8	74.1	91.2
TSM [84]		16	74.7	-
NL I3D [145]		32	74.9	91.6
Slow [37]		8	74.9	91.5
SlowFast [37]		4+32	75.6	92.1
RNL TSM (ours)		8	75.6	92.3
RNL TSM (ours)		16	77.2	93.1
RNL TSM_{En} (ours)		8+16	77.4	93.2
NL I3D [145]		128	76.5	92.6
NL I3D [145]	ResNet101	128	77.7	93.3
SlowFast [37]		16+64	<u>78.9</u>	<u>93.5</u>
LGD-3D RGB [102]		128	**79.4**	**94.4**

deeper backbone networks or longer clips as training inputs would consistently result in better performance in comparison with shallower backbone networks. On Kinetics, we employ

the shallower network ResNet50 as the backbone, and the length of our input video clips is a few times shorter than for the other methods, yet the results of our method are highly competitive with that of the other listed approaches which employ stronger backbones. On

Table 4.8 Comparisons with SOTA on Something-Something V1.

Model	Backbone	Frame×Clip×Crop	Top-1	Top-5
I3D [146]		192=32×2×3	41.6	72.2
NL I3D [146]		192=32×2×3	44.4	76.0
NL I3D + GCN [146]		192=32×2×3	46.1	76.8
TSM [84]		8=8×1×1	45.6	74.2
TSM [84]	ResNet50	16=16×1×1	47.2	77.1
TSM$_{En}$ [84]		24=(8+16)×1×1	49.7	78.5
RNL TSM (ours)		8=8×1×1	47.3	-
RNL TSM (ours)		16=16×1×1	49.4	-
RNL TSM$_{En}$ (ours)		24=(8+16)×1×1	51.3	80.6
SmallBig [80]		48=8×2×3	48.3	78.1
SmallBig [80]		96=16×2×3	50.0	79.8
SmallBig$_{En}$ [80]	ResNet50	144=(8+16)×2×3	51.4	80.7
RNL TSM (ours)		48=8×2×3	49.5	78.4
RNL TSM (ours)		96=16×2×3	51.0	80.3
RNL TSM$_{En}$ (ours)		144=(8+16)×2×3	52.7	81.5
RNL TSM (ours)	ResNet101	48=8×2×3	50.8	79.8
RNL TSM$_{En}$ (ours)	R101 + R50	144=(8+16)×2×3	**54.1**	**82.2**

Something-Something V1, when using ResNet50 as the backbone, the ensemble version of our model, the RNL TSM$_{En}$, using {8, 16} frames as inputs, achieves a higher accuracy than other approaches, w.r.t., single-clip & center-crop (Top-1: 51.3%) and multi-clip & multi-crop (Top-1: 52.7%). When adopting ResNet-101 as the backbone, we gain extra performance boost (Top-1: 50.8% vs. 49.5%). Moreover, the ensemble version of RNL TSM$_{En}$ (R101+R50) shown in the last row in Table 4.8 achieves the best accuracy (Top-1: 54.1%). All these results further demonstrate the effectiveness and efficiency of the proposed method.

4.7 Discussion

The experimental results show that by embedding either RNL or CPA modules into off-the-shelf CNNs, we achieve very strong global context modeling abilities, but RNL has shown higher performance than CPA in terms of classification accuracy. We have not yet deeply investigated what reason caused the accuracy gap, but from the evidence shown in our

experiments, the self-attention mechanism family might be a better way to model long-range dependencies. Nevertheless, self-attention mechanisms are computationally intensive, even though our RNL uses channel-wise convolution to reduce the computation. Due to the high computational demand of the self-attention mechanisms, their employment is limited to a network's deeper layers with smaller feature sizes. However, embedding the self-attention blocks into shallower network layers should perform better in principle, as suggested in [145]. On the contrary, the proposed CPA module which captures long-range dependencies with larger kernels is extremely lightweight, given the reason that each convolution filter in the CPA module has only one channel. The CPA module could be employed to shallower layers to get better results.

4.8 Conclusion

In this chapter, we have presented the region-based non-local (RNL) operation and the Convolution Pyramid Attention (CPA) module. The RNL operation is a novel self-attention mechanism that can effectively capture long-range dependencies by exploiting pair-wise region relationships. The proposed CPA module is a lightweight attention mechanism that leverages multiple convolutions of different kernel sizes, and each convolution has only a single channel, showing high efficiency when modeling global context. By connecting the RNL or CPA module with the Squeeze-and-Excitation (SE) module [59], we design the elegant vision chain, where the SE module learns channel attention while the RNL or CPA module learns spatio-temporal attention. We demonstrate that the RNL and CPA blocks are backbone-agnostic, which can be easily embedded into the off-the-shelf CNNs architectures.

We have performed ablation studies to investigate the effectiveness of the proposed RNL and CPA in various settings. To verify the efficiency and effectiveness of the proposed methodologies, we have conducted experiments on two video benchmarks, Kinetics400 and Something-Something V1. Our RNL outperforms the baseline and other recently proposed attention methods. Compared with RNL, CPA has lower accuracy. Nevertheless, the proposed CPA can improve the network performance with negligible additional computation, which is preferable in the deployments of real-world applications that require low latency. In addition, the attention maps of neural networks can be used to identify failed models. We have provided the attention map visualization for measuring the success of the proposed RNL and CPA methods. The attention maps of RNL show more accurate positional relationships than that of NL [145]. The attention maps of the network that has CPA modules embedded provide better salient region localization than the network without

CPA modules. All these results prove the effectiveness of the proposed attention mechanisms.

One limitation of our work in this chapter is that due to the limited compute nodes we can access, we have not investigated networks' lower layers for embedding attention blocks. Inserting attention blocks in lower layers could possibly generate higher accuracy by allocating more computations. In addition, the potentiality of the proposed RNL has not been fully explored, due to the fact that our work is built upon existing network architectures. The proposed attention mechanisms can also be used to design a completely new attention-based network.

Chapter 5

Busy-Quiet Video Disentangling for Efficient Video Understanding

5.1 Introduction

Over the last two chapters, we have studied methods for video representation learning as well as spatio-temporal attention mechanisms for global context modeling in the spatio-temporal domain. In this chapter, we start exploring video network learning from a distinctive direction: efficiently processing with low data redundancy thus resulting in high effectiveness.

Recent methods [36, 84, 101, 127, 128, 151] increase the efficiency of 3D CNNs by reducing the redundancy in the model parameters. However, these methods have ignored another important factor that causes the heavy computation in video processing: natural video data contains substantial redundancy in space and time. A video network can improve its efficiency by avoiding processing this redundant information repeatedly. Meanwhile, the effectiveness of 3D CNN [16] can be further improved by simply replacing raw time-series of RGB frame inputs with motion representations, such as TV-L1 flow [157]. We intend to design a unified architecture that extracts motion representations of videos and separate the redundant information simultaneously, and thus, to improve the network from both efficiency and effectiveness aspects. A rich representation of the information in video data can be realized by means of frequency analysis. Fine motion details from the boundaries of moving regions are characterized by high frequencies in the spatio-temporal domain. Meanwhile, lower frequencies are encoded with coarse information containing substantial redundancy, which causes low efficiency for those video models that take as input raw RGB frames. The study from this chapter proposes to disentangle video data into Busy and Quiet components,

in order to enable a computationally efficient and effective action recognition.

Video data processing can benefit from decomposing the video data into different streams of information and by allocating the appropriate computational resources to each of the streams. The Busy component of video information, describing fast-changing motion happening at the boundary of moving regions, is crucial for defining different actions in videos. The Quiet component, which is the counterpart of Busy, describes smooth background textures whose neighboring locations share similar information, being redundant in the video data processing. In our method, we firstly disentangle a video into Busy and Quiet components: Busy information of videos is conveyed within a specific spatio-temporal frequency bandwidth, which can be separated from the Quiet information by utilizing an end-to-end trainable Motion Band-Pass Module (MBPM). Subsequently, we separately process the Busy and Quiet components of videos, by allocating high-complexity processing for the Busy information while low-complexity processing is used for the Quiet information. The Busy-Quiet disentangling is optimized to the relevant movement following training with the videos from the dataset. As presented in Fig. 5.1, the Busy information characterizes the boundaries of the regions where fast movement happens. By applying the MBPM to a video of 3 segments, the number of representative frames is reduced from 9 to 3 whilst retaining and compressing the essential motion representation. Our experiments demonstrate that by simply replacing the RGB frame input with the motion representation extracted by MBPM, the performance of existing video models can be significantly boosted. Secondly, we design a two-pathway multi-scale architecture, called the Busy-Quiet Net (BQN), whose processing pipeline is shown in Fig. 5.2. The Busy pathway is responsible for processing the information distilled by the MBPM, representing fast changing information in the video. The other pathway, called Quiet, is devised to process the small changing information encoded with global smoothing spatio-temporal structures. In order to fuse the information from different pathways, we design Band-Pass Lateral Connection (BPLC) modules, which are set up between the layers of Busy and Quiet pathways. During the experiments, we demonstrate that BPLC modules represent the key factor to the overall model optimization success.

Compared with the frame summarization approaches [8, 136, 61], MBPM retains the strict temporal order of the frame sequences, which is considered essential for long-term temporal relation modeling. Compared with optical flow-based motion representation methods [157, 33, 63, 143, 158], the motion representation captured by MBPM has a smaller temporal size and can be employed on the fly. Meanwhile, efficient video models such as Octave Convolution [18], bL-Net [17] and SlowFast networks [37] only reduce the input redundancy

along either the spatial or temporal dimensions. Instead, the proposed BQN reduces the redundancy in the joint spatio-temporal space.

Our contributions from this chapter can be summarized as follows:

- A novel Motion Band-Pass Module (MBPM) is proposed for Busy motion information distillation. The new motion cue extracted by the MBPM significantly reduces temporal redundancy.
- We design a two-pathway Busy-Quiet Net (BQN) that separately processes the Busy and Quiet information in videos. After separating the Busy information using MBPM, we then can safely downsample the Quiet information to further decrease redundancy.
- Extensive experiments demonstrate the superiority of the proposed BQN over a wide range of models on four standard video benchmark datasets: Kinetics400 [16], Something-Something V1 [48], UCF101 [114] and HMDB51 [73]. The code is available at: *https://github.com/guoxih/busy-quiet-net*.

The rest of the chapter is organized as in the following: In Section 5.2, we give a brief overview of video feature learning. The proposed Motion Band-Pass Module (MBPM) and its training is described in Section 5.3. The Busy-Quiet Net (BQN) is presented in Section 5.4. The experimental results are provided in Section 5.5 and the conclusions of this chapter are drawn in Section 5.6.

5.2 Overview of Video Feature Learning

5.2.1 Enforcing Low Parameter Redundancy

Inception [120] and ResNet [54] are the concrete milestones in manual network design, which significantly alleviate the training difficulty of very deep networks by introducing multi-scale Inception blocks and establishing skip connections between layers in different depths. Their pretrained models on ImageNet have been widely employed for different downstream tasks. However, the ResNet and Inception families [54, 55, 150, 120, 121, 119] are heavily parameterized [103, 31]. For example, an instantiation of ResNet could contain tens or hundreds of millions of parameters. A video model using ResNet or Inception as the backbone would introduce non-negligible overhead, which could be an order of magnitude or more bigger than the image model based on ResNet. Given the progress in GPU performance, many methods [16, 50, 124, 126] tend to exploit the computationally intensive 3D convolution which allows simultaneous spatio-temporal data processing. The high capacity of ResNet enabled by over-parameterization is the key to high accuracy in large-scale human-curated

image datasets such as ImageNet. However, most human-curated video datasets are of small or medium-scale (*e.g*. the number of labeled samples of ImageNet are five times more than Kinetics400 [16]). Oquab *et al*. [97] pointed out that these high capacity networks tend to overfit on such small datasets when training from scratch. Recently published works [127, 36] remark that their lightweight networks have the accuracy on a par with those over-parameterized networks such as ResNet and Inception while having fewer parameters and lower computational requirements. It raised a concern that such an over-parameterized network may not be necessary for spatio-temporal modeling in videos, considering that the temporal features are of importance but their dimensionality usually is much smaller than the spatial features, and has less capacity requirement. Meanwhile, some studies focus on improving the efficiency of 3D CNN, such as Pseudo-3D Residual Net (P3D) [101], R(2+1)D [128], Separable 3D CNN (S3D) [151], Temporal Shift Module (TSM) [84], Channel-Separated Convolutional Network (CSN) [127] and Expand 3D (X3D) [36]. Our methods work on reducing redundancy in input space, which is complementary to the above approaches, but we intend to investigate the model that has low redundancy in both parameter (channel) space and input space. In our study, we show that Busy-Quiet Net (BQN) presents the characteristics of low parameter and low information redundancy by adopting efficient networks with fewer channels as its backbone.

5.2.2 Motion Representation

Two-stream model [112] and its variants [143, 40] achieve appearance and motion disentanglement by taking RGB frames and the optical flow as inputs of two CNNs, which achieves higher results than any of its single stream versions. Meanwhile, it implies that the separate processing of different types of features might make the network optimization more efficient. FlowNet series [63, 29] improve the optical flow estimation by using deep learning. However, optical flow estimation is inefficient with respect to memory storage and computation. To estimate the optical flow on the fly, some studies [95, 164] attempt to integrate optical flow estimation and action recognition into an end-to-end training framework. More recently, Optical Flow guided Feature (OFF) [118], TVNet [33], Flow-of-Flow [100] and other methods employing fast motion feature learning have been proposed. Some motion representations such as Squeezed Image and Dynamic Image summarize both the static and dynamic visual information of videos by utilizing Temporal Squeeze Pooling [61] and Approximate Rank Pooling [8], respectively. These methods work well provided that there is no severe camera shaking. Otherwise, the resulted squeezed images are blurred, resulting in poor discrimination between the moving objects and background. Compared to these approaches, our MBPM

produces higher accuracy while requiring less computation. Moreover, the proposed MBPM is rather like a basic component, which can be embedded with various video architectures.

5.2.3 Enforcing Low Information Redundancy

Enabled by deep learning, Big-Little Net (bL-Net) [17] adopts a downsampling strategy operating at the block level aiming to reduce the spatial redundancy of its feature maps. Subsequently, two branches are employed to separately process the feature maps with different resolutions. Chen *et al*. [18], replaced normal convolution operations with an Octave Convolution operation decomposing the video information into low-frequency and high-frequency components, while capturing more global information. For action recognition, the Big-Little-Video-Net bLVNet [34] extends the idea of bL-Net [17] to the temporal dimension. SlowFast networks [37] introduce two pathways, and decompose the input into the Slow and Fast components along the temporal dimension for efficient temporal modeling. However, the generalization of SlowFast to existing CNN architectures is poor, as it requires specifically customized CNNs as its backbones. Different from the existing methods, which only reduce feature redundancy either in the spatial dimensions or in the temporal dimension, the proposed BQN reduces the feature redundancy in the joint spatio-temporal space. We introduce a predefined trainable filter module, MBPM, to disentangle the video into Busy and Quiet components. In opposition to SlowFast, BQN architecture provides excellent generalization when considering any of existing CNNs as its backbone.

5.3 Motion Band-Pass Module (MBPM)

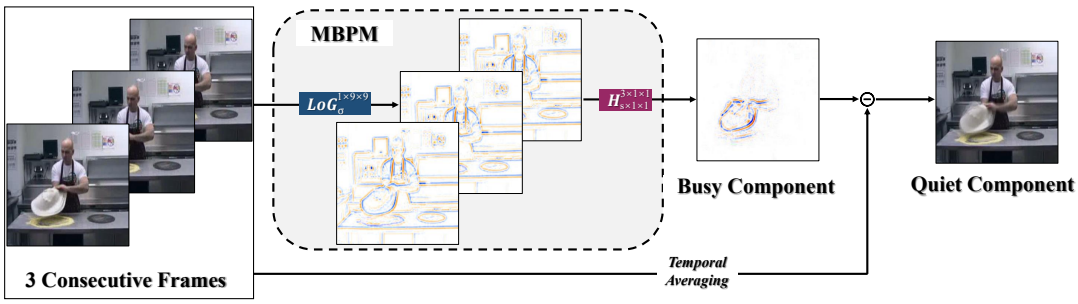


Fig. 5.1 The Motion Band-Pass Module (MBPM) disentangles a short frame sequence into Busy and Quiet components. For every three consecutive RGB frames, the MBPM generates a single-frame output, substantially reducing the redundancy.

The defining information required for action recognition consists of the moving objects and regions that make up the scene as well as their interactions. Particularly, the regions from the boundaries of the moving regions outlining both their shape and movement characteristics are essential for video representation. Actually, the movement of a moving object or region is characterized by its boundaries and how these are changing over time. On the other hand, the interior of rigidly moving objects, as well as their background, contains information which is rather constant in time. In this study we develop a mechanism for separating the information from regions critical in defining the movement in video, which are called Busy, from the regions which do not change much and are redundant in successions of frames, called Quiet. On one hand such a separation would enable a better characterization of the movement leading to better classification. On the other hand we will be able to allocate appropriate computational resources for the efficient computation of both the Busy and Quiet components.

In the following we introduce the Motion Band-Pass Module (MBPM), as a trainable 3D band-pass filter, which can distill the video information conveyed within a specific spatio-temporal frequency bandwidth, into Busy and Quiet information. A video clip can be defined as a function with three arguments, $\boldsymbol{I}^{(t)}(x,y)$, where x, y indicate the spatial dimensions, while $t = 1,\ldots,T$ is the temporal dimension and T represents the number of frames. The value of $\boldsymbol{I}^{(t)}(x,y)$ corresponds to the RGB pixel values at position (x,y) in t-th frame in the video. When considering the multi-channel video case, we repeat the same procedure for each feature channel, which for the first processing layer corresponds to the color components[1]. For the frequency band processing, we consider a filter based on the time differentials of the Laplacian of Gaussian applied in the spatial frame data. The output $\boldsymbol{\Gamma}$ of the frequency band-selection filter is given by:

$$\begin{aligned}\boldsymbol{\Gamma}(x,y,t) &= \frac{\partial^2}{\partial t^2}\left[\boldsymbol{I}^{(t)}(x,y) * LoG_{\sigma}(x,y)\right], \\ &\approx \sum_{t-1\leq i\leq t+1} h_{(i)} \cdot [\boldsymbol{I}^{(i)}(x,y) * LoG_{\sigma}(x,y)], \\ h_{(i)} &= \begin{cases} \frac{2}{3} & \text{if} \quad i = t, \\ -\frac{1}{3} & \text{otherwise}, \end{cases}\end{aligned} \tag{5.1}$$

[1] Specific notation is omitted for the sake of simplification.

for $t = 1, \ldots, T$ and '$*$' represents the convolution operation. Meanwhile, $LoG_\sigma(x,y)$ is a two-dimensional Laplacian of Gaussian with the scale parameter σ :

$$LoG_\sigma(x,y) = \nabla^2 G_\sigma(x,y) = -\frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]. \tag{5.2}$$

In Eq. (5.1), the second derivative with respect to t is numerically approximated by finite differences, literally implemented by the function $h_{(i)}$. The scale parameter σ of $LoG_\sigma(x,y)$ determines what information in which frequency bandwidth would be disentangled from video streams and consequently plays a crucial role in defining the Busy information. $LoG_\sigma(x,y)$ with a larger σ captures smoother textures of videos, and is therefore more robust to noise. On the other hand, a smaller σ would reliably capture some high frequency information characterizing fast moving objects.

From equations (5.1) and (5.2) we can observe that the 3D filtering function is fully-differentiable. In order to make the 3D band-pass filtering compatible with CNNs, we approximate it with two sequential channel-wise convolutional layers [106], as shown in Fig. 5.1. We name the discrete approximation implementation as the Motion Band-Pass Module (MBPM) which can be expressed in a computational form as :

$$\mathbf{\Gamma} \approx \text{MBPM}(\boldsymbol{I}) = H^{3\times1\times1}_{s\times1\times1}(LoG^{1\times k\times k}_\sigma(\boldsymbol{I})), \tag{5.3}$$

where $LoG^{1\times k\times k}_\sigma$ is referred to as a spatial channel-wise convolutional layer [106] with a $k \times k$ kernel, each channel of which is initialized with a Laplacian of Gaussian distribution of scale σ. The sum of kernel values is normalized to 1. Meanwhile, $H^{3\times1\times1}_{s\times1\times1}$ is referred to as a temporal channel-wise convolutional layer with a temporal stride s. In each channel, the kernel value of $H^{3\times1\times1}_{s\times1\times1}$ is initialized with $[-\frac{1}{3}, \frac{2}{3}, -\frac{1}{3}]$, which is a high-pass filter. In order to adjust to the specifics of the motion characteristics from the video frame sequences, the kernel parameters of MBPM are fine-tuned by training on the video datasets. We embed the MBPM in the CNN training process to form an end-to-end training pipeline, which is optimized with the classification loss. This training will result in an optimized MBPM, for disentangling the Busy information, defined by the characteristics of real videos.

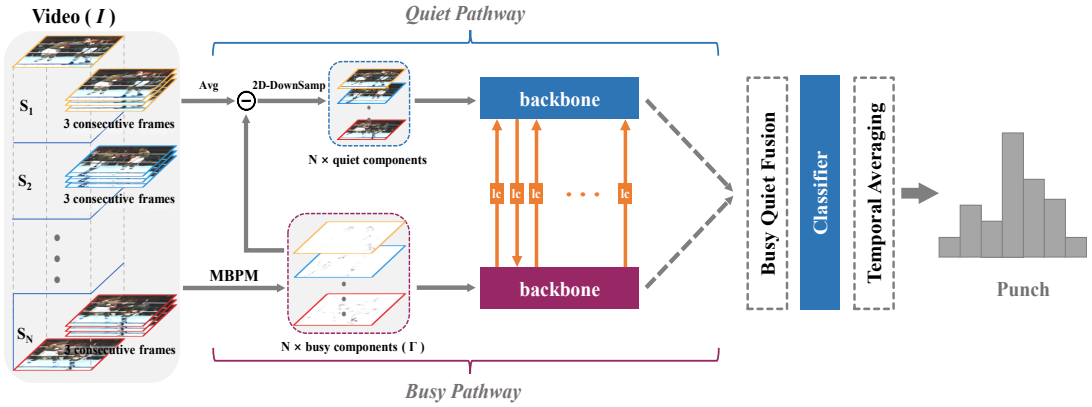


Fig. 5.2 The Busy-Quiet Net (BQN) is made up of two parallel pathways: Busy and Quiet. 'lc' indicates Band-Pass Lateral Connection. The MBPM firstly disentangles the input video into Busy and Quiet components. Subsequently, the backbone network from the Busy pathway processes the Busy information, while the network from the Quiet pathway processes the Quiet information. The outputs of the two pathways are eventually fused, and the final prediction is obtained by averaging the prediction scores across multiple segments.

5.4 Busy-Quiet Net (BQN)

The BQN architecture, illustrated in Fig. 5.2, contains two different processing pathways: one for processing the Busy information and another for the Quiet information. Splitting the processing into the two different processing pathways is justified by the fact that computational resources should be allocated differently, according to the characteristics of the information to be processed. The separation of the pathways is enabled the MBPM, whose construction was explained in the previous section. Meanwhile, the Busy and Quiet pathways are bridged by multiple Band-Pass Lateral Connections ("lc" in Fig. 5.2). These lateral connections enable information fusion between the two processing pathways at various processing stages.

5.4.1 The Busy pathway

The Busy pathway is designed to learn essential fine-grained movement features, such as those characterizing the transitions of distinct regions of movement. It takes as input the information filtered by the MBPM, corresponding to a specific spatio-temporal frequency band, selected following the training of MBPM, as described in Section 5.3. The stride of $H_{s\times1\times1}^{3\times1\times1}$ from Eq. (5.3) is set in the experiments to $s = 3$, which means that for every three consecutive RGB frames, MBPM generates an one-frame output. The MBPM output preserves the temporal order within the video while significantly reducing the redundant temporal information. For extracting more distinct moving object textures as well as of

transitional movement variation patterns, we would consider larger spatial input sizes for the Busy pathway.

5.4.2 The Quiet pathway

The Quiet pathway focuses on processing Quiet information, representing the characteristics of large regions of movement, such as the movement happening in the plain-textured background regions or from the inner regions of large moving objects. Such information is usually repeating itself from frame to frame and contains a lot of redundant information. These regions would require reduced computational processing for video characterization. The input to the Quiet pathway is considered to be the complementary of the MBPM output:

$$\text{2D-DownSamp}(\text{Avg}_{3\times1\times1}^{3\times1\times1}(\boldsymbol{I}) - \boldsymbol{\Gamma}), \tag{5.4}$$

where $\text{Avg}_{3\times1\times1}^{3\times1\times1}$ is temporal average pooling with a stride of 3 in the experiments, and $\boldsymbol{\Gamma}$ defines the Busy information, according to Eq. (5.3). We also perform bilinear downsampling in the spatial domain, along x and y coordinates (*i.e.* 2D-DownSamp), to reduce the redundant spatial information shared by neighboring locations in the Quiet information. In Section 5.5.3, we explore the Quiet information significance on the overall performance of BQN.

5.4.3 Band-Pass Lateral Connection (BPLC)

In ResNet [54] and DenseNet [62], skip connections (or shortcut connections) alleviate the vanishing gradient problem by enabling feature fusion from lower layers to higher layers. Inspired by this, we include a novel Band-Pass Lateral Connection (BPLC) module in the BQN architecture. The BPLC module has an MBPM embedded for information selection. The BPLCs established between the two pathways, Busy and Quiet, provide a mechanism for information exchange, enabling an optimal fusion of the two video information components Busy and Quiet corresponding to different frequency bands. Different from the lateral connections in other approaches [37, 38, 40, 85], the BPLC, enabled by MBPM, performs feature fusion and feature selection simultaneously, which shows higher performance than other lateral connection designs, according to the experimental results. We denote the two inputs of BPLC from the i-th residual blocks in the Busy and Quiet pathways, as $\mathbf{x}_f^i$ and $\mathbf{x}_c^i$, respectively. For simplifying the notation, we assume that $\mathbf{x}_f^i$ and $\mathbf{x}_c^i$ are of the same size. When their sizes are different, we adopt bilinear interpolation to match them in size. The

outputs $\mathbf{y}_f^i$ and $\mathbf{y}_c^i$ for the Busy and Quiet, respectively, are given by

$$\begin{aligned}
\mathbf{y}_f^i &= \begin{cases} \mathrm{BN}(\mathrm{MBPM}(\mathbf{x}_c^i)) + \mathbf{x}_f^i & \text{if} \quad \mathrm{mod}(i,2) = 0, \\ \mathbf{x}_f^i & \text{otherwise}, \end{cases} \\
\mathbf{y}_c^i &= \begin{cases} \mathbf{x}_c^i & \text{if} \quad \mathrm{mod}(i,2) = 0, \\ \mathrm{BN}(\phi(\mathbf{x}_f^i)) + \mathbf{x}_c^i & \text{otherwise}, \end{cases} \\
& \qquad\qquad i = 1, 2, \ldots, B
\end{aligned} \tag{5.5}$$

where B denotes the number of residual blocks in the backbone network (regarded as the network with residual block designs in the experiments). $\phi(\cdot)$ is a linear transformation that can be implemented as a $1 \times 1 \times 1$ convolution, or alternatively, when the channel number is very large, as a bottleneck Multi-layer Perceptron (MLP) for reducing computation. BN indicates Batch Normalization [65], with the weights initialized to zero. For the MBPM in BPLC, the convolutional stride of $H_{s\times 1\times 1}^{3\times 1\times 1}$ from Eq. (5.3) is set to $s = 1$, maintaining the same temporal size.

The fusion direction of BPLC reverses alternatively back and forth, as indicated in Fig. 5.2, providing better communication for the two pathways than the unidirectional lateral connections in [37, 85] whose information fusion direction is fixed, always fusing the information from one pathway to the other. By default, we place a BPLC between the two pathways right after each pair of residual blocks. The MBPM embedded in BPLC acts as a soft feature selection gate, where only busy information from the Quiet pathway is allowed to flow to the Busy pathway during the information fusion process. This design gives the best performance according to our experiments. The exploration of various designs of lateral connections is analyzed in Section 5.5.3.

5.5 Experiments

In this section, we first introduce the datasets and implementation details. Then we conduct ablation studies to investigate the efficiency and effectiveness of our proposed methods and find the appropriate parameters for the proposed methods. Finally, we compare with the state-of-the-art.

5.5.1 Implementation details

We use ResNet50 (R50) with TSM [84], X3D-M [36] and MobileNetV2 [106] as the backbones of our models. When not specified otherwise, R50 was used as the backbone. Aside from X3D-M [36], the backbone networks are pretrained on ImageNet [22].

We evaluate our approach on challenging human activities datasets such as Something-Something V1 [48], Kinetics400 [16], UCF101 [114] and HMDB51 [73]. Most of the videos in Kinetics400 (K400), UCF101 and HMDB51 can be accurately classified by only considering their background scene information, while the temporal relation between frames is not really that important. Meanwhile, in Something-Something (SS) V1, many action categories are more vaguely defined and characterized by symmetrical movements (*e.g.* "Pulling something from left to right" and "Pulling something from right to left"). Discriminating these symmetric actions requires models with strong temporal modeling ability. Since Something-Something is widely used for evaluating temporal modeling effectiveness, we consider this dataset as central to investigate the proposed Busy-Quiet Net (BQN).

For training, we utilize the dense sampling strategy [145] for Kinetics400. As for other datasets, we utilize the uniform sampling strategy as shown in Fig. 5.2, where a video is equally divided into N segments, and 3 consecutive frames in each segment are randomly sampled to constitute a video clip of length $T = 3N$. Unless specified otherwise, a default video clip is composed of $N = 8$ segments with a spatial frame size of 224^2. We train our models on 16 or 64 GPUs (NVIDIA Tesla V100), using Stochastic Gradient Descent (SGD) with momentum 0.9 and cosine learning rate schedule. In order to prevent overfitting, we add a dropout layer before the classification layer of each pathway in the BQN model. Following the experimental settings in [84, 143], the learning rate and weight decay parameters for the classification layers are 5 times those of the convolutional layers. Meanwhile, we only apply L2 regularization to the weights in the convolutional and classification layers to avoid overfitting.

During testing, we sample a single clip per video with center cropping for efficient inference [84], which is used in our ablation studies. When pursuing high accuracy, we consider sampling multiple clips&crops from the video and then averaging the prediction scores of multiple spacetime "views" (spatial crops $\times$ temporal clips) as it was used in [37].

Hyperparameters for models based on ResNet

For the models on Kinetics400 [16], the initial learning rate, batch size, total epochs, weight decay and dropout ratio are set to 0.08, 512 (8 samples per GPU), 100, 2e-4 and 0.5, respectively. For Something-Something V1 [48], these hyperparameters are set to 0.12, 256, 50, 8e-4 and 0.8, respectively. We use linear warm-up [89] for the first 7 epochs to overcome early optimization difficulty. When fine-tuning the Kinetics models on UCF101 [114] and HMDB51 [73], we freeze all of the batch normalization [65] layers except for the first one to avoid overfitting, following the recipe in [143]. The initial learning rate, batch size, total epochs, weight decay and dropout ratio are set to 0.001, 64 (4 samples per GPU), 10, 1e-4 and 0.8, respectively.

Hyperparameters for models based on X3D-M

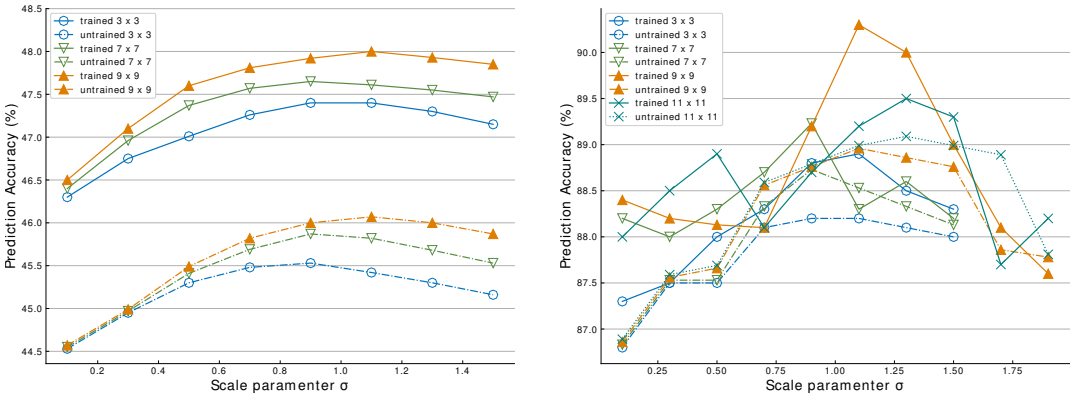
For the models on Kinetics400 [16], the initial learning rate, batch size, total epochs, weight decay and dropout ratio are set to 0.4, 256 (16 samples per GPU), 256, 5e-5 and 0.5, respectively. For Something-Something V1 (SS V1), the models trained from scratch use the followings hyperparameters: learning rate 0.2, batch size 256, total epochs 100, weight decay 5e-5 and dropout ratio 0.5. When fine-tuning the Kinetics models, the initial learning rate, batch size, total epochs, weight decay and dropout ratio are set to 0.12, 256 (16 samples per GPU), 60, 4e-4 and 0.8, respectively.

5.5.2 Ablation Studies for MBPM

In this section, we conduct ablation studies on multiple datasets to evaluate the best settings for the MBPM and BQN. We show top-1 and top-5 prediction accuracy (%), as well as computational complexity measured in GFLOPs for a single crop & single clip.

Instantiations and Settings.

The scale σ from Eq. (5.2) and the kernel size of the spatial channel-wise convolution $LoG_{\sigma}^{1\times k\times k}$ have a significant impact on the performance of MBPM. We vary the scale σ and the kernel size, while evaluating the prediction accuracy, to search for the optimal settings. Meanwhile, in order to highlight the importance of the MBPM training, we compare the performance when using the trained MBPM with that of an untrained MBPM, whose kernel weights are not optimized with the classification loss. The results on SS V1 are shown in



(a) SS V1 (b) UCF101

Fig. 5.3 Results on Something-Something V1 (SS V1) and UCF101 when varying the scale σ and kernel size $k \times k$ of the spatial channel-wise convolution in MBPM. The results are averages of multiple experiment runs.

Fig. 5.3(a). We summarize two facts: 1) the optimal value of σ for the MBPM changes as its kernel size changes, and the MBPM with $\sigma = 1.1$ and a spatial kernel size 9×9 gives the best performance within the searching range. 2) optimizing the parameters of MBPM with the classification loss generally produces higher prediction accuracy. In our preliminary work, we have verified that different datasets share the same optimal settings of MBPM. We search for the optimal settings of the scale σ and kernel size $k \times k$ of MBPM on UCF101 dataset. The results are presented in Fig. 5.3(b). We observe that the experimental results vary greatly under different settings. Nevertheless, the optimal scale is $\sigma = 1.1$ when setting the kernel size as 9×9, which is the same as that for the Something-Something V1 dataset. Furthermore, we try a larger kernel of 11×11, but then the results show a performance drop. We speculate that this is caused by insufficient training. In the following experiments, we set MBPM in the Busy pathway as trainable with the scale $\sigma = 1.1$ and the kernel size of 9×9, unless specified otherwise.

Efficiency and Effectiveness of the MBPM

We draw an apple-to-apple comparison between the proposed MBPM and other motion representation methods [157, 8, 33, 63, 143, 158]. The motion representations produced by these methods are used as inputs to the backbone network and the comparison results are shown in Table 5.1. We follow the experimental settings from [158] for a fair comparison. The backbone network used for all the methods is ResNet50 [54]. We use the computer code provided by the original authors for these methods to generate the network inputs. For any of these motion representations, we divide the representation of a video into 8 segments and

Table 5.1 MBPM *vs*. other motion representation methods, when training on UCF101, SS V1 and K400 datasets and considering ResNet50 [54] as backbone. The additional parameters to the backbone network and the computational complexity (in FLOPs) required by each method are reported. † denotes our reimplementation.

Representation Method	**Efficiency Metrics**		**UCF101**	**SS V1**	**K400**
	FLOPs	**#Param.**			
RGB	-	-	87.1	46.5	71.2
RGB Difference [143]	-	-	87.0	46.6	71.4
TV-L1 Flow [157]	-	-	88.5	37.4	55.7
Dynamic Image† [8]	-	-	86.2	43.4	68.3
FlowNetC† [63]	444G	39.2M	87.3	26.3	-
FlowNetS† [63]	356G	38.7M	86.8	23.4	-
TVNet† [33]	3.3G	0.2K	88.6	45.2	58.5
Persistent Appearance [158]	2.8G	1.1K	89.5	45.1	57.3
MBPM	0.3G	0.2K	**90.3**	**48.0**	**72.3**

randomly select one frame from each segment. Following the practices used in the Temporal Segment Network (TSN) [143] and the Persistent Appearance Network (PAN) [158], the output activation of 8 segments is averaged for the final prediction score. In our reimplementation, Dynamic Image [8] generates one dynamic image for every 6 consecutive RGB frames, which consumes the same number of RGB frames as Persistent Appearance [158]. Our MBPM generates one representative frame for every 3 consecutive RGB frames. As for TVNet [33] and TV-L1 Flow [157], a one-frame input to the backbone network is formed by stacking 5 frames of the estimated flow along the channel dimension, which requires 6 RGB frames. All models are pretrained on ImageNet. For Something-Something V1 and Kinetics400, we use the hyperparameters specified in Section 5.5.1 to train all models. For UCF101, we set the initial learning rate, batch size, total epochs, weight decay and dropout ratio to 0.01, 64 (4 samples per GPU), 80, 1e-4 and 0.5, respectively. According to the results from Table 5.1 the proposed MBPM outperforms all other motion representation methods by big margins, while its computational cost is negligible, which strongly demonstrates the high efficiency and effectiveness of the MBPM. Moreover, we consider different input modalities, and the two-stream fusion of "RGB+MBPM" has higher accuracy than the fusion of "RGB+Flow," according to the results shown in Table 5.2.

In order to visually observe the differences between the outputs of our MBPM and other motion representation methods, in Fig. 5.4, we show some example video frames and their corresponding motion representations generated by different methods. The optical flow estimates the instantaneous velocity and direction of movement in every position, where the color represents the direction of movement while the brightness represents the absolute value of instantaneous velocity in a position. In contrast, TVNet [33], Persistent Appearance [158] and MBPM are more absorbed in the visual information presented in boundary regions where motion happens. In Fig. 5.4, the textures captured by TVNet, Persistent Appearance and the MBPM do not present obvious differences. However, our MBPM is more simple and requires less computation.

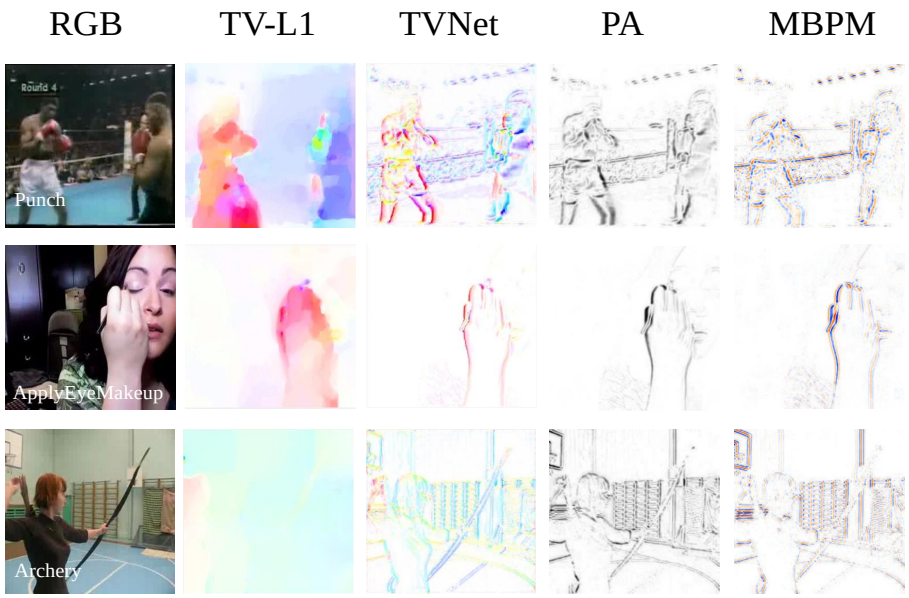


Fig. 5.4 Comparison between visualizations of different motion representations, TV-L1 Flow, TVNet, Persistent Appearance (PA) and the proposed MBPM on the UCF101 dataset after training. TV-L1 Flow [157] evaluates the movement in every spatial position, while TVNet [33], PA [158] and our MBPM, capture the outline of the moving objects.

Generalization to different CNNs used as backbones

ResNet [54] significantly alleviates the training difficulty of very deep networks by establishing skip connections between layers at different depths. In the previous section, we have seen that the proposed MBPM can efficiently and effectively increase the prediction

Table 5.2 Using different CNNs as backbones on SS V1. ResNet50 and MobileNetV2 have TSM [84] embedded.

CNN backbone	Modality	Pretrain	#Segments (N)	Accuracy
ResNet50 [54]	RGB	ImageNet	8	46.5
	RGB+Flow			49.8
	MBPM			48.0
	RGB+MBPM			**50.3**
MobileNetV2 [106]	RGB	ImageNet	8	38.7
	MBPM			39.8
X3D-M [36]	RGB	None	16	45.5
	MBPM			46.9

accuracy of ResNet on multiple video datasets. However, the main problem in ResNet is its over-parameterization: an instantiation of ResNet could contain tens or hundreds of millions of parameters, resulting in the cycle of the training and inference being too long. In order to shorten the model training and inference cycle, we attempt to employ our method to the efficient network architectures, MobileNetV2 [106] and X3D [36]. We further show that the proposed MBPM is a generic plug-and-play unit that is compatible with a wide range of network architectures. By simply placing an MBPM after the input layers of these off-the-shelf neural networks, their performance on action recognition tasks has non-trivial improvement. In Table 5.2 we present the results of ResNet50 [54], MobileNetV2 [106] and X3D-M [36] before and after being configured with the MBPM. MobileNetV2 [106] is a computation-efficient CNN, which is specifically tailored for mobile devices, allowing fast deployment in resource-constrained environments. After being configured with our MBPM, MobileNetV2 shows a higher motion feature learning ability, increasing its accuracy by 1.1% on Something-Something V1. Additionally, we adapt our method to X3D-M, an instantiation of the efficient video network design, namely X3D, which is a 3D expansion of the core concept of channel-wise separable convolution in MobileNet [58], using the progressive network expansion approach [36]. Notably, the MBPM improves X3D-M by 1.4%, even though the original X3D-M network has been specially designed for spatio-temporal feature learning. The steady performance improvements of these existing networks when embedding the MBPM have sufficiently evidenced the high generalization ability of MBPM. In Table 5.9 and Table 5.10 we also present the results of the BQN architectures that employ different types of backbone network. More details about the analysis of the generalization ability of BQN over different types of backbone network is given in Section 5.5.4.

5.5.3 Ablation Studies for Busy-Quiet Net (BQN)

BQN *vs*. Quiet+Busy.

In order to evaluate the architecture effectiveness, we compare the proposed BQN with the simple fusion (Quiet+Busy), which mimics the two-stream model [112], by averaging the predictions of the two pathways trained separately. The results from Table 5.3 indicate that the simple fusion of two individual pathways (Quiet+Busy) generates higher top-1 accuracy (50.3%) than the individual pathways, which indicates that the features learned by the Quiet and Busy pathways are complementary. BQN has 51.6% top-1 accuracy, which is 1.3% better than the fusion, Quiet+Busy. The high-performance gain strongly demonstrates the advantages of the proposed BQN architecture.

Table 5.3 Complementarity of Quiet and Busy. "Quiet" and "Busy" refer to that the Quiet and Busy pathways are trained separately.

Model	Top-1 (%)	Top-5 (%)	GFLOPs
Quiet	46.5	75.3	32.8
Busy	48.0	76.8	32.8
Quiet+Busy	50.3	79.0	65.7
BQN	**51.6**	**80.5**	65.9

Fusion strategies

In the following we evaluate the effectiveness of different approaches for fusing the predictions of the Busy and Quiet pathways in the BQN architecture. The results of using different fusion methods are shown in Table 5.4. We observe that the average fusion gives the best result among the listed approaches, which is in line with the experimental results of the two-stream model in [40]. The approach of concatenation fusion is second only to the averaging. Moreover, we observe that placing the average fusion layer after the fully-connected (fc) layer is better than placing it before.

Effectiveness of the BPLC

We can set a maximum of up to 16 BPLCs in the BQN architecture when using TSM R50 [84] as the backbone. ResNet50 [54] contains four stages, named res2, res3, res4, res5, respectively. These stages are composed of 3, 4, 6, 3 residual blocks, respectively. For the BPLCs in the stages res2, res3 and res4, we set the spatial kernel size of MBPM as 7×7, and the scale $\sigma = 0.9$. As for stage res5, whose feature size is relatively small, the kernel

Table 5.4 Fusion Strategies. The fully-connected (fc) layers of the two pathways share their parameters

Fusion Method	**Position**	**Top-1 (%)**	**Top-5 (%)**
Averaging	before fc	50.9	79.8
Averaging	after fc	**51.6**	**80.5**
Max	after fc	50.1	78.7
Concatenation	before fc	51.3	80.2

size is therefore set to 3×3. Table 5.5, illustrates that adding BPLCs to all processing stages leads to improved performance. From Table 5.6, we can observe that the model performance improves gradually as the number of BPLCs increases. The substantial performance gains demonstrate the importance of using BPLCs for BQN.

Table 5.5 Adding BPLCs to various processing stages of ResNet50 backbone. In each stage, we set one BPLC after its first residual block.

Stages	**No. of BPLC**	**Top-1 (%)**	**Top-5 (%)**
res2	1	49.8	79.1
res2,res3	2	50.1	78.7
res2,res3,res4	3	50.2	79.0
res2,res3,res4,res5	4	50.2	79.2

Table 5.6 The effect of the number of BPLCs.

No. of BPLCs	**Top-1 (%)**	**Top-5 (%)**	**GFLOPs**
0	49.6	78.9	65.7
4	50.2	79.2	65.8
8	50.7	79.7	65.8
16	**51.6**	**80.5**	65.9

Lateral Connection (LC) Designs

In order to illustrate the rationality of the proposed BPLC design, we compare it with other LC designs. The diagrams of different LC designs are illustrated in Fig. 5.5, where LC-I and LC-II are unidirectional, and LC-III is bidirectional. Table 5.7 shows that the bidirectional design LC-III has higher accuracy than the unidirectional designs LC-I and LC-II. Among the listed designs, the proposed BPLC, which reverses the information fusion direction back and forth alternatively, provides the highest accuracy. We also compare the BPLC with LC-V

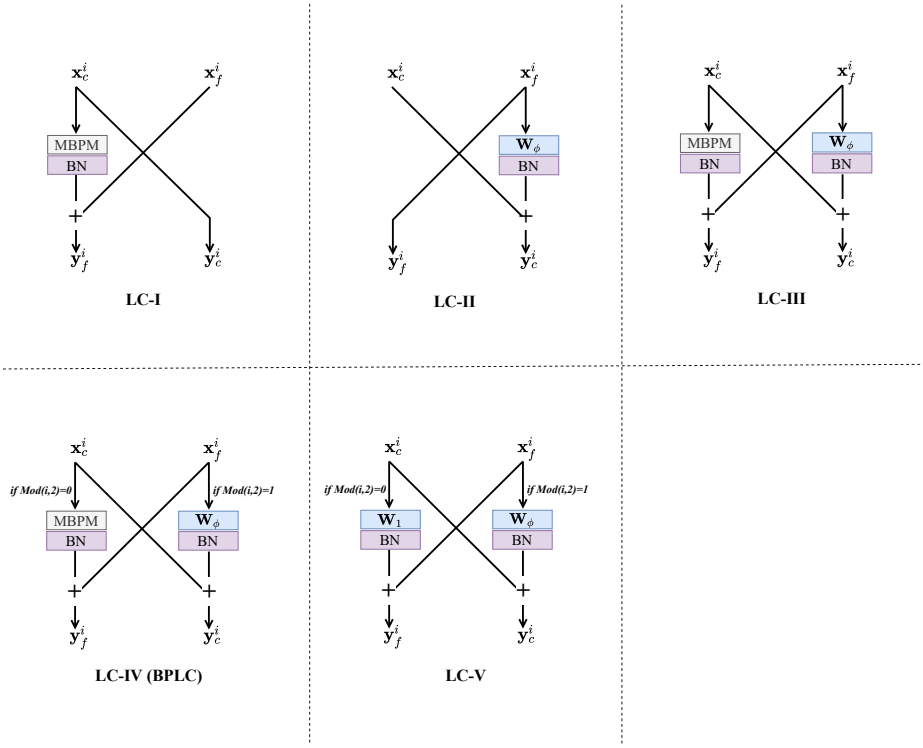


Fig. 5.5 Diagrams of various lateral connection (LC) designs. Bilinear interpolation is used for resizing the feature maps when $\mathbf{x}_c^i$ and $\mathbf{x}_f^i$ do not have the same spatial size. i refers to the index of the residual block. $\mathbf{W}_\phi$ and $\mathbf{W}_1$ denote the weights of the linear transformation.

that does not contain an MBPM. As a result, LC-V shows lower accuracy than the BPLC, which demonstrates the importance of embedding MBPM in the BPLC.

Table 5.7 Various LC designs. 16 LCs are set in the BQN.

Design	**Top-1(%)**	**Top-5 (%)**
LC-I	50.9	79.8
LC-II	50.9	79.7
LC-III	51.5	80.2
BPLC (LC-IV)	**51.6**	80.5
LC-V	51.3	79.9

Spatial-temporal input size

In BQN, the Busy pathway takes as input the MBPM output, which has the same spatial size as the raw video clip, while the temporal size is one-third of the length of the raw video clip. Meanwhile, the Quiet pathway takes as input the complementary component of the

MBPM output, expressed through Eq. (5.4). Table 5.8 shows that with the same temporal size of 8 for the inputs, the spatial size combination of 160^2 and 256^2 for the Quiet and Busy, respectively, has slightly better top-1 accuracy (+0.1%) than the combination of 224^2 and 224^2 but saves 5.2 GFLOPs in computational cost. We also attempt to reduce the temporal input size of the Quiet pathway. However, this would result in a performance drop. One possible explanation is that due to the temporal average pooling in the Quiet pathway, the input's temporal size is already reduced to one-third of the raw video clip. An even smaller temporal size could fail to preserve the correct temporal order of the video, and therefore harms the temporal relation modeling.

Table 5.8 Effect of the spatio-temporal input size. The input size is formatted as (width^2 $\times$ time).

Input size for Quiet	**Input size for Busy**	**Top-1 (%)**	**Top-5 (%)**	**GFLOPs**
$224^2 \times 8$	$224^2 \times 8$	51.6	80.5	65.9
$192^2 \times 8$	$224^2 \times 8$	51.5	79.9	58.0
$160^2 \times 8$	$224^2 \times 8$	51.3	80.1	50.5
$128^2 \times 8$	$224^2 \times 8$	50.7	79.2	44.4
$224^2 \times 8$	**$256^2 \times 8$**	**51.8**	**80.5**	**77.1**
$192^2 \times 8$	$256^2 \times 8$	51.7	80.2	68.3
$160^2 \times 8$	**$256^2 \times 8$**	**51.7**	**80.5**	**60.7**
$128^2 \times 8$	$256^2 \times 8$	51.3	79.4	54.6
$160^2 \times 6$	$256^2 \times 8$	49.6	78.3	55.5
$224^2 \times 4$	$224^2 \times 8$	48.7	77.1	49.4

5.5.4 Comparison with the State-of-the-Art

We compare BQN with current state-of-the-art methods on the four datasets. In BQN, the Quiet and Busy pathways' spatial input size is set to 160^2 and 256^2, respectively.

Results on Something-Something V1

Table 5.9 summarizes the comprehensive comparison, including the inference protocols, corresponding computational costs (FLOPs) and the prediction accuracy. Our method surpasses all other methods by good margins. For example, the multi-clip accuracy of BQN$_{24f}$[2] with TSM R50 is 7.2% higher than NL I3D GCN$_{32f}$ [146] while requiring 5$\times$ fewer FLOPs. Among the models based on ResNet50, BQN$_{48f}$ has the highest top-1 accuracy

[2]The subscript 24f indicates that video clips of 24 frames are used for experiments.

Table 5.9 Results on Something-Something V1. "N/A" indicates the numbers are not available. † denotes our reimplementation. **Bold** and underline show the highest and second highest results, respectively.

Method	pre-train	Backbone	Frames×Crops×Clips	FLOPs	#Param.	Top-1 (%)	Top-5 (%)
NL I3D GCN [146]	ImageNet	3D R50	32×3×2	303G×3×2	62.2M	46.1	76.8
$ECO_{EnLite\ RGB+Flow}$ [165]		Inc+3D R18	(92+552)×1×1	N/A	300M	49.5	-
TSN [143]		R50	8×1×1	33G×1×1	-	19.7	46.6
$TRN_{RGB+Flow}$ [162]		BNInception	(8+48)×1×1	N/A	36.6M	42.0	-
TSM_{En} [84]		R50	(16+8)×1×1	98G×1×1	48.6M	49.7	78.5
$TSM_{RGB+Flow}$ [84]		R50	(16+96)×1×1	N/A	48.6M	52.6	81.9
TEA [81]		R50	16×3×10	70G×3×10	24.4M	52.3	81.9
TDN [142]		R101	16×1×1	132G×1×1	-	55.3	83.3
SmallBig [80]		-	16×3×2	105G×3×2	-	50.0	79.8
bLVNet-TAM [34]		bLR50	8×1×2	12G×1×2	25M	46.4	76.6
GTA_{En} [52]		TSM R50	(16+8)×3×2	-	-	<u>56.5</u>	<u>83.1</u>
PAN_{Full} [158]		TSM R50	40×1×2	67.7G×1×2	-	50.5	79.2
PAN_{En} [158]		TSM R50	(40+40)×1×2	134G×1×2	-	53.4	81.1
PAN_{En} [158]		TSM R101	(40+40)×1×2	251G×1×2	-	55.3	82.8
ir-CSN [127]	None	3D R101	32×1×10	73.8G×1×10	22.1M	48.4	-
ir-CSN [127]		3D R152	32×1×10	96.7G×1×10	-	49.3	-
TSM R50 [84]	ImageNet	R50	16×1×1	65G×1×1	24.3M	47.2	77.1
BQN		TSM R50	24×1×1	60G×1×1	47.4M	51.7	80.5
BQN		TSM R50	24×3×2	60G×3×2	47.4M	53.3	82.0
BQN		TSM R50	48×3×2	121G×3×2	47.4M	54.3	82.0
BQN		TSM R101	48×3×2	231G×3×2	85.4M	54.9	81.7
X3D-M† [36]	None	-	16×3×2	6.4G×3×2	3.3M	46.7	75.5
BQN	None	X3D-M	48×3×2	9.7G×3×2	6.6M	50.6	79.2
BQN	K400	X3D-M	48×3×2	9.7G×3×2	6.6M	53.7	81.8
$\mathbf{BQN}_{En}$	ImageNet + K400	TSM R101 +X3D-M	(48+48)×3×2	241G×3×2	92M	**57.1**	**84.2**

(54.3%), which surpasses the second-best, TEA_{16f} [81], by a margin of +2%. Furthermore, our signal-clip BQN_{24f} has higher accuracy (51.7%) than most other multi-clip models, requiring only 60 GFLOPs. By adopting a deeper backbone (TSM R101), BQN_{48f} has 54.9% top-1 accuracy, higher than any other model.

Table 5.10 Comparison results on Kinetics400. We report the inference cost of multiple "views" (spatial crops × temporal clips). † denotes our reimplementation.

Method	**Backbone**	**Frames × views**	**FLOPs**	**Top-1 (%)**	**Top-5 (%)**
bLVNet-TAM [34]	bLR50	16×9	561G	72.0	90.6
TSM [84]	R50	16×30	2580G	74.7	-
TEINet [87]		16×30	2580G	76.2	92.5
TEA [81]		16×30	2100G	76.1	92.5
STM [68]		16×30	2010G	73.7	91.6
X3D-M† [36]	-	16×30	186G	75.1	92.2
VoV3D-M [79]	-	16×30	172G	74.7	92.1
$\text{SlowFast}_{4\times16}$ [37]	3D R50	32×30	1083G	75.6	92.1
CorrNet [135]	3D R50	32×10	1150G	77.2	-
R(2+1)D [128]	R34	32×10	1520G	72.0	91.4
ip-CSN [127]	3D R101	32×30	2490G	76.7	92.3
SlowFast+GTA [52]	3D R101	-	4110G	**79.8**	**94.1**
SmallBigNet [80]	R101	32×12	6552G	77.4	93.3
TSN [143]	BNInception	25×10	530G	69.1	88.7
PAN_{Full}	TSM R50	40×2	176G	74.4	91.6
I3D_{RGB} [16]	Inception V1	64×N/A	N/A	71.1	89.3
Oct-I3D [18]	-	N/A×N/A	N/A	74.6	-
NL I3D [145]	3D R101	128×30	10770G	77.7	93.3
BQN	TSM R50	48×10	1210G	76.8	92.4
BQN	TSM R50	72×10	1820G	77.3	93.2
BQN	X3D-M	48×30	291G	77.1	92.5

Although the BQN architecture has produced decently high classification accuracy on the dataset, the additional computational cost introduced by BQN is not negligible when the employed backbones have high parameter redundancy. For example, the computational cost of BQN_{48f} with TSM R50 for a single clip is 121 GFLOPs, which is almost 2 times more than that of TSM R50. If BQN adopts a deeper ResNet as the backbone, its computational complexity would be further increased, which could lead to high-latency in its real-world applications. When using X3D-M as the backbone, BQN achieves the ultimate efficiency, possessing very low redundancy in both feature channel and spatio-temporal dimensions. BQN with X3D-M processes 4× more video frames than vanilla X3D-M, with only 50%

Table 5.11 Results on HMDB51 and UCF101. We report the mean class accuracy over the three official splits.

Method	Backbone	HMDB51	UCF101
StNet [53]	R50	-	93.5
TSM [84]	R50	73.5	95.9
STM [68]	R50	72.2	96.2
TEA [81]	R50	73.3	96.9
DI Four-Stream [8]	ResNeXt101	72.5	95.5
TVNet [33]	BNInception	71.0	94.5
$\text{TSN}_{\text{RGB+Flow}}$ [143]	BNInception	68.5	94.0
$\text{I3D}_{\text{RGB+Flow}}$ [16]	3D Inception	**80.7**	**98.0**
PAN_{Full} [158]	TSM R50	77.0	96.5
BQN	TSM R50	<u>77.6</u>	<u>97.6</u>

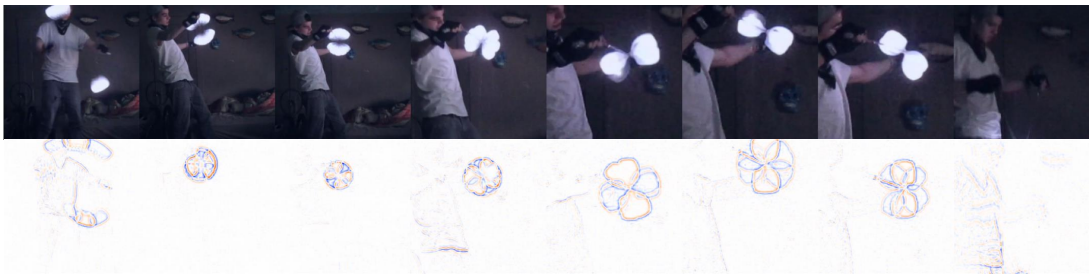
additional FLOPs. Compared with TSM R50_{16f}, BQN with X3D-M trained from scratch produces 3.4% higher top-1 accuracy with the computational complexity of 14% of TSM R50_{16f}. The ensemble version $\textbf{BQN}_{\text{En}}$ achieves the state-of-the-art top-1/5 accuracy (57.1%/84.2%).

Results on Kinetics400, UCF101 and HMDB51

Table 5.10 shows the comparison results on Kinetics400. For a fair comparison, we only list the models with the spatial input size of 256^2. BQN_{72f} with TSM R50 achieves 77.3%/93.2% top-1/5 accuracy, which is better than the 3D CNN-based architecture, I3D [16], by a big margin of +6.2%/3.9%. When BQN uses TSM R50 or X3D-M as its backbone, it consistently shows higher accuracy than $\text{SlowFast}_{4\times16}$. Particularly, BQN with X3D-M has 1.5% higher top-1 accuracy than $\text{SlowFast}_{4\times16}$, while requiring 3.7× fewer FLOPs. Meanwhile, BQN_{72f} with TSM R50 is 2.7% better than Oct-I3D [18] for top-1 accuracy. The results on two smaller datasets, UCF101 and HMDB51, are shown in Table 5.11, where we report the mean class accuracy over the three official splits. We pretrain our model on Kinetics400 to avoid overfitting. The accuracy of our method is calculated using the inference protocol (3 crops×2 clips). BQN with TSM R50 outperforms most other methods except for $\text{I3D}_{\text{RGB+Flow}}$, which uses an additional optical flow input modality.

5.5.5 Visualization Analysis

In Figures 5.6-(1) and (2), we visualize two complex human activities from Kinetics 400, representing "Spinning poi" and "Chopping wood", respectively, where the top row of images are uniformly selected from the original video while underneath we show the correspond-



(1) Spinning poi

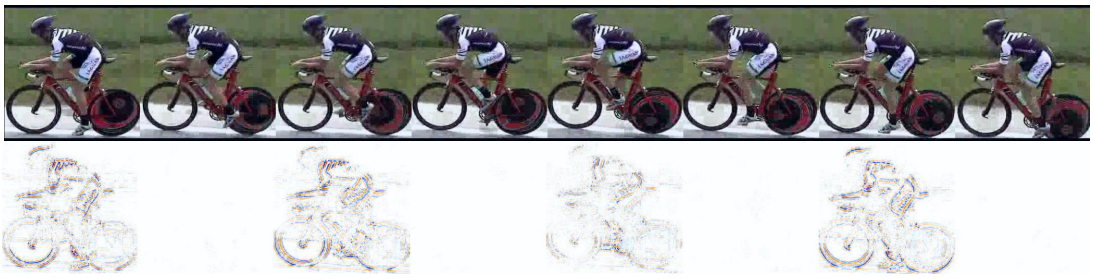
(2) Chopping wood

Fig. 5.6 Example videos and their corresponding MBPM outputs from Kinetics 400.

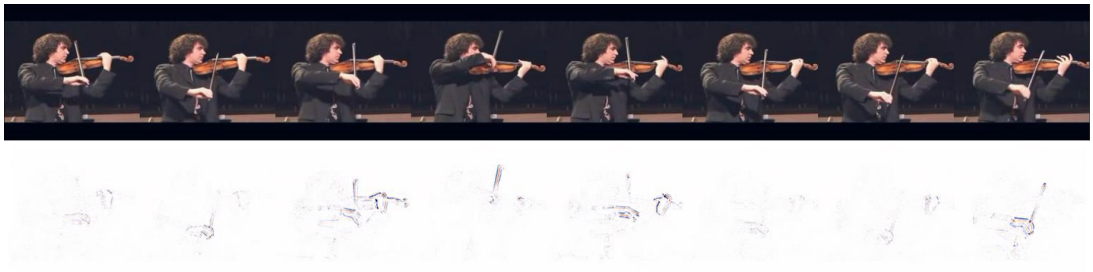
(1) Pretending to take something out of something

(2) Pretending to turn something upside down

Fig. 5.7 Example Videos and their corresponding MBPM outputs from Something-Something V1.

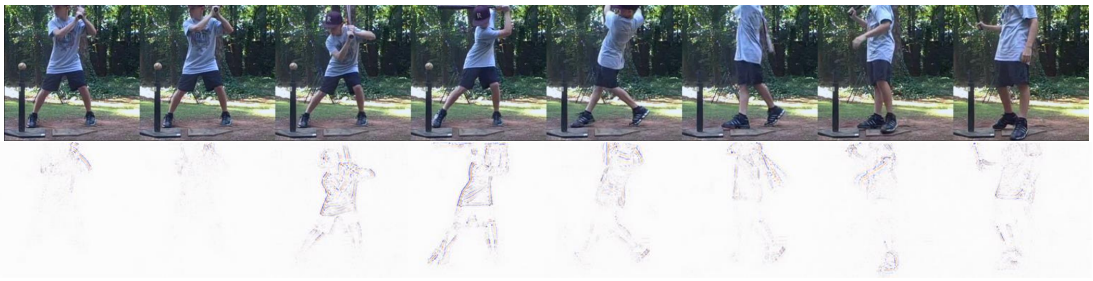


(1) Biking



(2) Playing violin

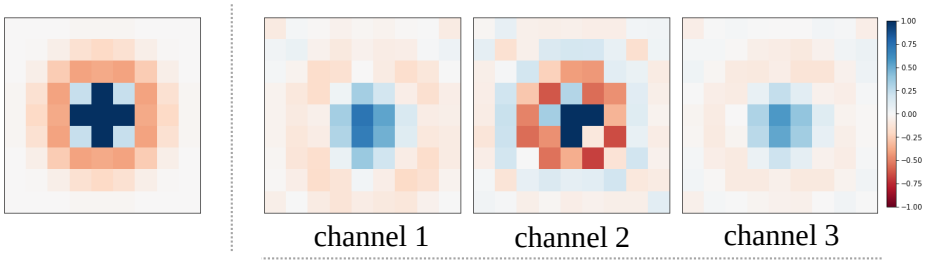
Fig. 5.8 Example Videos and their corresponding MBPM outputs from UCF101.



(1) Swing baseball

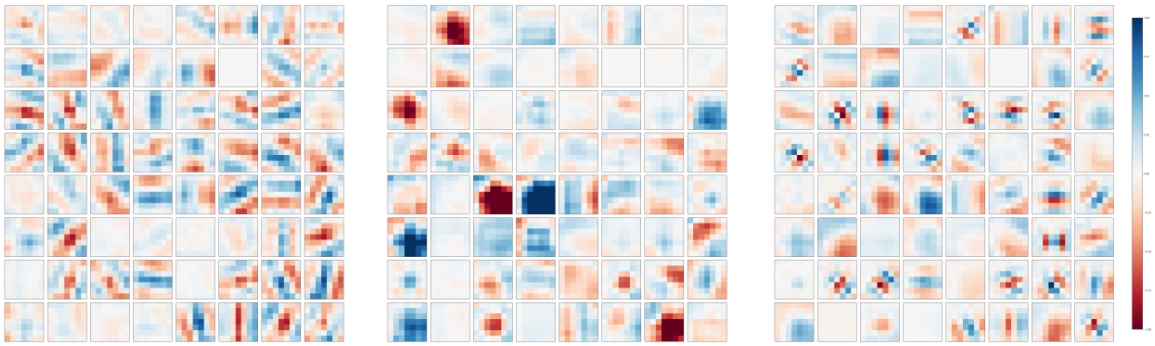
(2) Kick

Fig. 5.9 Example Videos and their corresponding MBPM outputs from HMDB51.



(a) Untrained LoG (b) Trained LoG

Fig. 5.10 Visualization of the spatial channel-wise convolution $LoG_{\sigma}^{1\times k\times k}$ of MBPM in the Busy pathway before and after training on Kinetics400. The 9×9 channel-wise convolution is initialized with a Laplacian of Gaussian with the scale parameter $\sigma = 1.1$. Best viewed in color and zoomed in.



(a) Busy pathway (b) Quiet pathway (c) TSM ResNet50

Fig. 5.11 Visualization of the first channels of the 64 conv1 filters of BQN after training on Kinetics400. All 64 filters have a size of 7×7. From left to right, in (a), (b) and (c), we respectively present the trained conv1 filters in the Busy pathway, Quiet pathway and TSM ResNet50. We observe that the kernels of the 64 filters in the Busy pathway display stripe-like shapes, consistent with band-pass filters, while those for the filters in the Quiet pathway are more like larger blobs. The conv1 in TSM ResNet50 (baseline) contains both types of filters from the Busy and Quiet pathways. Best viewed in color and zoomed in.

ing busy stream selected by an MBPM. In Figures 5.7-(1) and (2) we show examples of Busy streams for complex activities characteristic to Something-Something V1 dataset. In Figures 5.8-(1) and (2) we present the visualization results of the MBPM output selection for "Biking" and "Playing violin" video frame sequences from UCF101. In Figures 5.9-(1) and (2) we present the visualization results of the MBPM output selection for "Swing baseball" and "Kick" video frame sequences from HMDB51. For a better representation of the movement output we use the optical flow visualization approach from [6] to visualize the busy stream selected by MBPM. We can observe from these examples that the extracted representations are stable when jittering and other minor camera movements are present in the videos. MBPM not only that suppresses the stationary information and the background movement, but it also highlights the boundaries of moving objects and regions, which are of vital importance for action discrimination. For example, in the "spinning poi" video, showing the movement of a illuminating object from Fig. 5.6-(1), MBPM highlights well the poi's movement rather than the movement of the background or the performer.

In Fig. 5.10, we visualize the kernel of the spatial convolution $LoG_{\sigma}^{1\times k\times k}$ of MBPM in the Busy pathway. Interestingly, before and after training, kernels always present a similar shape to a Mexican hat function. In Fig. 5.11, we visualize the first channel of the 64 filters in the first layers of the BQN and the baseline (TSM ResNet50). We can observe that the Busy and Quiet pathways' filters have quite distinct shapes for their kernels, suggesting that the Busy and Quiet pathways learned different types of features after training.

5.6 Conclusion

In this chapter, we have presented the idea of busy-quiet video disentangling for efficient video data processing in action recognition tasks. For this aim, we propose the Motion Band-Pass Module (MBPM) which, following training, defines different spatio-temporal frequency bands for the Busy and Quiet information in the video data. The Busy information separated by the MBPM provides important motion cues for action recognition, such as those characterizing the regions of movement transitions or the boundaries of moving objects in videos. Enabled by MBPM, we design an efficient and effective two-stream spatio-temporal processing architecture called Busy-Quiet Net (BQN), to separately process Busy and Quiet video data information. Meanwhile, we have presented the novel Band-Pass Lateral Connection (BPLC) module that enables simultaneously performing feature selection as well as feature fusion between the Busy and Quiet pathways in BQN. The proposed two-stream video processing network, besides disentangling the video information for better recognition,

allows for a better allocation of the computational resources, where more processing power is used for the Busy stream and less for the Quiet. We have demonstrated that by adopting efficient networks of low parameter redundancy, BQN has culminated in very low redundancy in both parameter and input space, allowing ultra-efficiency. We have conducted extensive ablation experiments to evaluate the design rationality of the proposed MBPM, Band-Pass Lateral Connection and BQN.

Our work has a few limitations. First of all, the low parameter redundancy in the BQN architecture relies on the lightweight of the two backbone networks whose designs need additional research effort. In addition, the two pathways in BQN must use two structurally identical networks. In principle, the Busy pathway is more significant than the Quiet pathway, so its backbone should be heavier than the Quiet pathway. Regardless of that, the current BQN architecture has shown higher efficiency and effectiveness than recent state-of-the-art models in multiple video benchmarks. The proposed Busy and Quiet video disentanglement can also be used for video analysis in various applications.

Chapter 6

Conclusion

This chapter summarizes the main contributions of this thesis. Discussions about the limitations, as well as potential directions for future work are also provided.

6.1 Contributions

The research work reported in this thesis develops several novel machine learning algorithms for efficient and effective spatio-temporal modeling for video action recognition.

In Chapter 3, we developed two novel methods for high discriminative video representation extraction. In the first half of Chapter 3, we proposed a compact video representation, termed Squeezed Image. Each squeezed image has three RGB channels outputted by the proposed Temporal Squeeze Pooling (TSP) module, which summarizes the dynamics of long video frame sequences. A standard CNN is capable of learning high-level spatio-temporal features by simply taking the squeezed images as the input. By embedding multiple TSP modules into a standard 2D CNN, we construct the Temporal Squeeze Network (TeSNet) architecture, in which the TSP modules gradually reduce the temporal size of feature maps saving computational cost. One fatal drawback of TSP is that it would shuffle the temporal order. This would significantly harm the performance of those datasets that rely on temporal relations to discriminate different actions.

In the second half of Chapter 3, we upgrade TSP to become a pixel-level operation, termed Pixel-Wise Temporal Projection (PWTP). PWTP is capable of separating motion-relevant information from static appearance in videos. The extracted representations present high-discriminative characteristics even for actions from similar scenes.

Chapter 4 develops two attention mechanisms for modeling long-term dependencies: the region-based non-local (RNL) operation and Convolution Pyramid Attention (CPA). The proposed RNL module is a self-attention mechanism capable of capturing globally spatio-temporal context by utilizing the pair-wise relationships between local neighboring regions. In comparison with the previous self-attention mechanism [145], our RNL demonstrated stronger robustness against noise while resulting in higher accuracy with a lower computational cost.

The proposed CPA module also presents an excellent performance for global context modeling by leveraging unusually large convolution kernels. Moreover, the CPA module has a better multi-scale feature learning ability than regular convolution blocks. In order to capture both spatio-temporal attention and channel-wise attention in a unified module, we link the SE block [59] with the RNL or CPA module to construct an attention chain. The attention chain is a backbone-agnostic module, that can be simply embedded into off-the-shelf networks and boost performance.

In Chapter 5, we present the concept of busy-quiet video disentangling for efficient spatio-temporal modeling in video recognition. We propose the Motion Band-Pass Module (MBPM), which is fully differentiable, to disentangle the video information into two components, termed Busy and Quiet. The Busy component encodes fine-grained motion features, essential for distinguishing different actions in videos, whilst the Quiet component encodes coarse-grained information with substantial redundancy. Then we propose the Busy-Quiet Net (BQN) architecture which has the MBPM embedded. BQN achieves efficient information processing by adopting an intelligent computational resources allocation strategy. Regarding the Busy information, the spatial resolution is enlarged in the input layer in order to capture more motion details. With reference to the Quiet component, the information is performed by downsampling to reduce redundancy. BQN presents extremely lightweight characteristics and demonstrates higher performance than other heavier architectures.

6.2 Future Work

The PWTP module described in Chapter 3 has it own optimization criteria. In order to train the PWTP module and its corresponding backbone CNN in an end-to-end manner, we have introduced two joint training algorithms. Recently, more advantageous jointing algorithms such as the one from [153] have been proposed. As future work, we could adapt these algorithms to optimize our model. The recent Vision transformer [28] (ViT) architecture

have demonstrated higher performance than CNNs in some visual applications. Although the potential of ViT has yet to be fully explored, we can attempt to combine our representation learning methods with ViT to evaluate the generalization ability of our methods further. We have done some experiments that use ViT as the backbone architecture to build our models, and we have seen performance improvement, but a thorough evaluation is still necessary to conduct.

In Chapter 4, we proposed the self-attention mechanism RNL for global context modeling. As future work, we can upgrade RNL to be a multi-head self-attention mechanism [131]. Furthermore, considering that the self-attention mechanism is the base unit of many Transformer architectures, we can attempt to design an efficient transformer by using a multi-head version of RNL.

For efficient video data processing, in the BQN architecture described in Chapter 5, the largest spatial size in the Busy pathway is set to 256^2. However, the published works [36, 37] suggest that using a larger spatial input size could further increase accuracy. Meanwhile, the spatial size of the Quiet pathway can be made smaller. Relevant experiments are needed to conduct in the future.

The topic of multimodal deep representation learning for video recognition [93] has attracted increasing research attention recently. The research work of this thesis has not taken the audio modality into consideration. By considering the additional audio input, the performance of video recognition could be further improved. What is more, research work that combines self-supervised learning and multimodal representation learning [92, 2] could be a promising direction in the future.

References

[1] M. Ahad, T. Ogata, J. Tan, H. Kim, and S. Ishikawa. Motion recognition approach to solve overwriting in complex actions. In *International Conference on Automatic Face & Gesture Recognition*, pages 1–6. IEEE, 2008.

[2] H. Akbari, L. Yuan, R. Qian, W. Chuang, S. Chang, Y. Cui, and B. Gong. VATT: Transformers for multimodal self-supervised learning from raw video, audio and text. *Advances in Neural Information Processing Systems (NIPS)*, 34, 2021.

[3] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6836–6846, 2021.

[4] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39. Springer, 2011.

[5] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR), arXiv preprint arXiv:1409.0473*, 2015.

[6] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[7] G. Bertasius, H. Wang, and L. Torresani. Is space-time attention all you need for video understanding. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[8] H. Bilen, B. Fernando, E. Gavves, and A. Vedaldi. Action recognition with dynamic image networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 40(12):2799–2813, 2018.

[9] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3034–3042, 2016.

[10] A. Bobick and J. Davis. An appearance-based representation of action. In *Proceedings of International Conference on Pattern Recognition (ICPR)*, volume 1, pages 307–312, 1996.

[11] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(3):257–267, 2001.

[12] G. Bradski and J. Davis. Motion segmentation and pose recognition with motion history gradients. *Machine Vision and Applications*, 13(3):174–184, 2002.

[13] M. Bregonzio, S. Gong, and T. Xiang. Recognising action as clouds of space-time interest points. In *Proceedings IEEE Conference Computer Vision Pattern Recog. (CVPR)*, pages 1948–1955. IEEE, 2009.

[14] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 60–65. IEEE, 2005.

[15] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCV-w)*, 2019.

[16] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.

[17] C. Chen, Q. Fan, N. Mallinar, T. Sercu, and R. Feris. Big-little net: An efficient multi-scale feature representation for visual and speech recognition. *arXiv preprint arXiv:1807.03848*, 2018.

[18] Y. Chen, H. Fan, B. Xu, Z. Yan, Y. Kalantidis, M. Rohrbach, S. Yan, and J. Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3435–3444, 2019.

[19] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1251–1258, 2017.

[20] N. Crasto, P. Weinzaepfel, K. Alahari, and C. Schmid. Mars: Motion-augmented rgb stream for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7882–7891, 2019.

[21] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *Proceedings of the European conference on computer vision (ECCV), vol. LNCS 3952*, pages 428–441, 2006.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on*

Computer Vision and Pattern Recognition (CVPR), pages 248–255, 2009.

[23] J. Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

[24] A. Diba, A. Pazandeh, and L. Van Gool. Efficient two-stream motion and appearance 3d cnns for video classification. *arXiv preprint arXiv:1608.08851*, 2016.

[25] A. Diba, V. Sharma, and L. Van Gool. Deep temporal linear encoding networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2329–2338, 2017.

[26] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72, 2005.

[27] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015.

[28] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations (ICLR), arXiv preprint arXiv:2010.11929*, 2021.

[29] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2758–2766, 2015.

[30] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 726–733, 2003.

[31] A. El-Nouby, G. Izacard, H. Touvron, I. Laptev, H. Jegou, and E. Grave. Are large-scale datasets necessary for self-supervised pre-training? *arXiv preprint arXiv:2112.10740*, 2021.

[32] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6824–6835, 2021.

[33] L. Fan, W. Huang, C. Gan, S. Ermon, B. Gong, and J. Huang. End-to-end learning of motion representation for video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6016–6025, 2018.

[34] Q. Fan, C. Chen, H. Kuehne, M. Pistoia, and D. Cox. More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2264–2273, 2019.

[35] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Proceedings IEEE Conference Computer Vision Pattern Recog. (CVPR)*, volume 2, pages 524–531. IEEE, 2005.

[36] C. Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 203–213, 2020.

[37] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*,

pages 6202–6211, 2019.

[38] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3468–3476, 2016.

[39] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal multiplier networks for video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4768–4777, 2017.

[40] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016.

[41] B. Fernando, P. Anderson, M. Hutter, and S. Gould. Discriminative hierarchical rank pooling for activity recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1924–1932, 2016.

[42] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5378–5387, 2015.

[43] B. Fernando and S. Gould. Learning end-to-end video classification with rank-pooling. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1187–1196. PMLR, 2016.

[44] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *Proceedings ISPRS intercommission conference on fast processing of photogrammetric data*, volume 6, pages 281–305. Interlaken, 1987.

[45] H. Gammulle, S. Denman, S. Sridharan, and C. Fookes. Two stream lstm: A deep fusion framework for human action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 177–186. IEEE, 2017.

[46] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 971–980, 2017.

[47] M. Goodale and A. Milner. Separate visual pathways for perception and action. *Trends in neurosciences*, 15(1):20–25, 1992.

[48] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, volume 1, pages 5842–5850, 2017.

[49] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang. Transformer in transformer. *Advances in Neural Information Processing Systems (NIPS)*, 34, 2021.

[50] K. Hara, H. Kataoka, and Y. Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6546–6555, 2018.

[51] C. Harris, M. Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[52] B. He, X. Yang, Z. Wu, H. Chen, S. Lim, and A. Shrivastava. GTA: Global temporal attention for video action understanding. *arXiv preprint arXiv:2012.08510*, 2020.

[53] D. He, Z. Zhou, C. Gan, F. Li, X. Liu, Y. Li, L. Wang, and S. Wen. Stnet: Local and global spatial-temporal modeling for action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8401–8408, 2019.

[54] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[55] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 630–645. Springer, 2016.

[56] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[57] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computer*, 9(8):1735–1780, 1997.

[58] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[59] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018.

[60] M. Hu. Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8(2):179–187, 1962.

[61] G. Huang and A. G. Bors. Learning spatio-temporal representations with temporal squeeze pooling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[62] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017.

[63] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2462–2470, 2017.

[64] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 448–456. PMLR, 2015.

[65] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML), vol. PMLR 37*, page 448–456, 2015.

[66] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proceedings IEEE Conference Computer Vision Pattern Recog. (CVPR)*, pages 3304–3311. IEEE, 2010.

[67] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(1):221–231, 2013.

[68] B. Jiang, M. Wang, W. Gan, W. Wu, and J. Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE/CVF International*

Conference on Computer Vision (ICCV), pages 2000–2009, 2019.

[69] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014.

[70] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[71] A. Klaser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 995–1004, 2008.

[72] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[73] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2556–2563, 2011.

[74] Z. Lan, Y. Zhu, A. Hauptmann, and S. Newsam. Deep local video feature for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, pages 1–7, 2017.

[75] I. Laptev. On space-time interest points. *Int. J. Computer Vision*, 64(2-3):107–123, 2005.

[76] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[77] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[78] M. Lee, S. Lee, S. Son, G. Park, and N. Kwak. Motion feature network: Fixed motion filter for action recognition. In *Proceedings of the European conference on computer vision (ECCV)*, pages 387–403, 2018.

[79] Y. Lee, H. Kim, K. Yun, and J. Moon. Diverse temporal aggregation and depthwise spatiotemporal factorization for efficient video classification. *IEEE Access*, 2021.

[80] X. Li, Y. Wang, Z. Zhou, and Y. Qiao. Smallbignet: Integrating core and contextual views for video classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1092–1101, 2020.

[81] Y. Li, B. Ji, X. Shi, J. Zhang, B. Kang, and L. Wang. Tea: Temporal excitation and aggregation for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 909–918, 2020.

[82] Y. Li, W. Li, V. Mahadevan, and N. Vasconcelos. Vlad3: Encoding dynamics of deep features for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1951–1960, 2016.

[83] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. Snoek. Videolstm convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 166:41–50, 2018.

[84] J. Lin, C. Gan, and S. Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7083–7093, 2019.

[85] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017.

[86] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[87] Z. Liu, D. Luo, Y. Wang, L. Wang, Y. Tai, C. Wang, J. Li, F. Huang, and T. Lu. Teinet: Towards an efficient architecture for video recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11669–11676, 2020.

[88] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

[89] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *International Conference on Learning Representations (ICLR), arXiv preprint arXiv:1608.03983*, 2017.

[90] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR), arXiv preprint arXiv:1711.05101*, 2019.

[91] C. Ma, M. Chen, K., and G. AlRegib. Ts-lstm and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition. *Signal Processing: Image Communication*, 71:76–87, 2019.

[92] P. Morgado, N. Vasconcelos, and I. Misra. Audio-visual instance discrimination with cross-modal agreement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12475–12486, 2021.

[93] A. Nagrani, S. Yang, A. Arnab, A. Jansen, C. Schmid, and C. Sun. Attention bottlenecks for multimodal fusion. *Advances in Neural Information Processing Systems (NIPS)*, 34, 2021.

[94] D. Neimark, O. Bar, M. Zohar, and D. Asselmann. Video transformer network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3163–3172, 2021.

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

[96] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *Proceedings of the European conference on computer vision (ECCV)*, pages 490–503. Springer, 2006.

[97] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1717–1724, 2014.

[98] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Computer Vision and Image Understanding*, 150:109–125, 2016.

[99] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *Proceedings of the European conference on computer vision (ECCV), vol LNCS 8693)*, pages 581–595, 2014.

[100] A. Piergiovanni and S. Ryoo. Representation flow for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9945–9953, 2019.

[101] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5533–5541, 2017.

[102] Z. Qiu, T. Yao, C.-W. Ngo, X. Tian, and T. Mei. Learning spatio-temporal representation with local and global diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12056–12065, 2019.

[103] I. Radosavovic, R. Kosaraju, R. Girshick, K. He, and P. Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10428–10436, 2020.

[104] S. Sadanand and J. J. Corso. Action bank: A high-level representation of activity in video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1234–1241, 2012.

[105] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *Int. Jour. of Comp. vision*, 105(3):222–245, 2013.

[106] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018.

[107] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of computer vision*, 37(2):151–172, 2000.

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

[109] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[110] O. Sener and V. Koltun. Multi-task learning as multi-objective optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 31. Curran Associates, Inc., 2018.

[111] Y. Shi, Y. Tian, Y. Wang, W. Zeng, and T. Huang. Learning long-term dependencies for action recognition with a biologically-inspired deep network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 716–725, 2017.

[112] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NIPS)*, pages 568–576, 2014.

[113] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR), arXiv preprint arXiv:1409.1556*, 2015.

[114] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

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

[116] L. Sun, K. Jia, K. Chen, D. Yeung, B. E. Shi, and S. Savarese. Lattice long short-term memory for human action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2147–2156, 2017.

[117] L. Sun, K. Jia, D. Yeung, and B. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4597–4605, 2015.

[118] S. Sun, Z. Kuang, L. Sheng, W. Ouyang, and W. Zhang. Optical flow guided feature: A fast and robust motion representation for video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1390–1399, 2018.

[119] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 4, page 12, 2017.

[120] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[121] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for comp. vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[122] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2820–2828, 2019.

[123] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning (ICML), vol. PMLR 97*, pages 6105–6114, 2019.

[124] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 140–153, 2010.

[125] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 10347–10357. PMLR, 2021.

[126] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[127] D. Tran, H. Wang, L. Torresani, and M. Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5552–5561, 2019.

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

[129] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE access*, 6:1155–1166, 2017.

[130] J. Van Gemert, J. Geusebroek, C. Veenman, and A. Smeulders. Kernel codebooks for scene categorization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 696–709. Springer, 2008.

[131] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

[132] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, 2017.

[133] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176, 2011.

[134] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3551–3558, 2013.

[135] H. Wang, D. Tran, L. Torresani, and M. Feiszli. Video modeling with correlation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 352–361, 2020.

[136] J. Wang, A. Cherian, F. Porikli, and S. Gould. Video representation learning using discriminative pooling. In *Proceedings of the IEEE/CVF Conference on Computer*

Vision and Pattern Recognition (CVPR), pages 1149–1158, 2018.

[137] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Proceedings IEEE Conference Computer Vision Pattern Recog. (CVPR)*, pages 3360–3367. IEEE, 2010.

[138] L. Wang, Y. Qiao, and X. Tang. Mining motion atoms and phrases for complex action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2680–2687, 2013.

[139] L. Wang, Y. Qiao, and X. Tang. Motionlets: Mid-level 3d parts for human motion recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2674–2681, 2013.

[140] L. Wang, Y. Qiao, and X. Tang. Video action detection with relational dynamic-poselets. In *Proceedings of the European conference on computer vision (ECCV), vol LNCS 869*, pages 565–580, 2014.

[141] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4305–4314, 2015.

[142] L. Wang, Z. Tong, B. Ji, and G. Wu. Tdn: Temporal difference networks for efficient action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1895–1904, 2021.

[143] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proceedings of the European conference on computer vision (ECCV), vol LNCS 9912*, pages 20–36, 2016.

[144] W. Wang, E. Xie, X. Li, D. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 568–578, 2021.

[145] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7794–7803, 2018.

[146] X. Wang and A. Gupta. Videos as space-time region graphs. In *Proceedings of the European conference on computer vision (ECCV)*, pages 399–417, 2018.

[147] Y. Wang, M. Long, J. Wang, and P. Yu. Spatiotemporal pyramid network for video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1529–1538, 2017.

[148] G. Willems, T. Tuytelaars, and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *Proceedings of the European conference on computer vision (ECCV), Vol. LNCS 5303*, pages 650–663, 2008.

[149] S. Woo, J. Park, J. Lee, and I. So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[150] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2017.

[151] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European conference on computer vision (ECCV), vol. LNCS 11219*, pages 305–321,

2018.

[152] L. Yeffet and L. Wolf. Local trinary patterns for human action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 492–497, 2009.

[153] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems (NIPS)*, 33:5824–5836, 2020.

[154] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z. Jiang, F. Tay, J. Feng, and S. Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 558–567, 2021.

[155] K. Yue, M. Sun, Y. Yuan, F. Zhou, E. Ding, and F. Xu. Compact generalized non-local network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6510–6519, 2018.

[156] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, 2015.

[157] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint Pattern Recognition Symposium, vol. LNCS 4713*, pages 214–223, 2007.

[158] C. Zhang, Y. Zou, G. Chen, and L. Gan. Pan: Persistent appearance network with an efficient motion cue for fast action recognition. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 500–509, 2019.

[159] H. Zhang, Y. Hao, and C. Ngo. Token shift transformer for video classification. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 917–925, 2021.

[160] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2018.

[161] Y. Zhao, Y. Xiong, and D. Lin. Trajectory convolution for action recognition. *Advances in Neural Information Processing Systems (NIPS)*, 31, 2018.

[162] B. Zhou, A. Andonian, A. Oliva, and A. Torralba. Temporal relational reasoning in videos. In *Proceedings of the European conference on computer vision (ECCV)*, pages 803–818, 2018.

[163] J. Zhu, B. Wang, X. Yang, W. Zhang, and Z. Tu. Action recognition with actons. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3559–3566, 2013.

[164] Y. Zhu, Z. Lan, S. Newsam, and A. Hauptmann. Hidden two-stream convolutional networks for action recognition. In *Asian Conference on Computer Vision*, pages 363–378. Springer, 2018.

[165] M. Zolfaghari, K. Singh, and T. Brox. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 695–712, 2018.

[166] B. Zoph, V. Vasudevan, J. Shlens, and Q. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8697–8710, 2018.