On Spatio-Temporal and Motion Feature Representation Learning with Deep Neural Network for Accurate Video Action Recognition

(正確なビデオアクション認識のための深層ニューラルネットワークを用いた時空間と動作特徴量の表現学習について)

January, 2023

Doctor of Philosophy (Engineering)

Yuri Yudhaswana Joefrie
ユリ　ユダスワナ　ジュフリ

Toyohashi University of Technology

Acknowledgements

I want to start by thanking the Almighty for His never-ending goodness in allowing me to mentally and physically complete this difficult endeavor.

Although this work is believed to be personal, it will only be possible to achieve the objectives with the assistance of other individuals. First and foremost, my heartfelt thanks go to Professor Masaki Aono. He has been my mentor, supervisor, and advisor, and it is a joy to express my sincere appreciation and gratitude. He has also been giving me the invaluable opportunity to research computer vision, one of the most excitingly progressing fields in computer science. He has been patient in discussing research problems and editing drafts with my limited English skills. His commitment, keen attention, and overwhelming desire to assist me were exclusively and primarily responsible for completing my task. His prompt guidance, rigorous investigation, expert counsel, and scientific approach have helped me achieve this journey.

I also thank committee members, particularly Professor Shigeru Kuriyama and Professor Norihide Kitaoka, for their insightful comments and suggestions. I would also like to thank the Toyohashi University of Technology office staff, particularly the International Affairs Division (IAD), for their ongoing assistance.

With the exceptional support of the Japan International Cooperation Agency (JICA), this study would have taken shape. JICA supports my Ph.D. through the JICA Innovative Asia scholarship program, where the author is an awardee of Batch 1. JICA provided me with funding throughout my study period in Japan and facilitated an internship experience in a Japanese private company.

I am also grateful to all members of the Knowledge Data Engineering (KDE) Laboratory, especially my tutor Nowshed-san, for helping me during my stay in Toyohashi. Also, I am much obliged to all Perhimpunan Pelajar Indonesia Toyohashi (PPITy) and the Indonesian community in Toyohashi for marvelous and unforgettable moments. I will remember the kindness and help they provide. I cannot neglect to mention Akhmad "Alioke" Alimudin for his support and help at the end of my doctoral study period. Furthermore, I'm also grateful to the entire Department of Information

Technology at Tadulako University. All my colleagues and staff provide me with good support and encouragement during my study.

A special thanks to my family. Words cannot express how grateful I am to my parents, parents-in-law, and relatives for all the sacrifices that they have made on my behalf. Their prayer for me was what sustained me thus far.

Lastly and importantly, I wish to thank my beloved wife, Shophia Mahardika, for her unfailing love and support throughout our marriage and for helping me finish my doctoral studies. I am also thankful to my children, Almairakan Ganesha Joefrie, Rasta Riuno Arziki, and Ranum Pradnya Iswari, for cheering me on when I was disheartened.

Finally, I would like to show my gratitude for the cultures, traditions, and Japanese mentality. I learn too many things from here.

Yuri Yudhaswana Joefrie
January 2023

Abstract

Accurate action recognition from digital videos is one of core and challenging computer vision problems that has been studied for decades. The potential applications of action recognition include video surveillance, human-computer interface, visual information retrieval, and unmanned driving. How to perform effective and efficient analysis of the video footage is vital given the recent exponential growth of surveillance data on the Internet. More and more individuals are becoming accustomed to posting photographs and phrases on social media sites like Instagram, Facebook, Twitter, and Flickr to express their feelings and ideas on almost any occasion or topic. As a result, studying the vast quantity of videos on social media has become increasingly important. Traditional machine learning techniques that only extract computable characteristics have limitations and need to function better with large amounts of visual input. However, convolutional neural networks (CNNs) and other deep learning techniques have significantly improved in this area.

Even though deep learning models exhibit such prominent results, the proposed work for action recognition still needs to improve, despite the numerous solutions that have been developed thus far. The model's ability to precisely detect changes in action variety, in addition to its ability to effectively recognize targets and actions from the background, presents a significant challenge. The conventional model will be hampered as a result of this issue.

Meanwhile, action recognition in videos can be divided into two tasks. The first task is to recognize a single action inside a video. The second task is to recognize various activities consecutively or simultaneously in each video at an arbitrary time. Each activity may have a unique set of features. Thus, different approaches are needed to deal with any issues that may develop. In this dissertation, we propose two approaches to tackle the challenges of action recognition and validate them in challenging datasets.

We tackle several issues regarding spatio-temporal and motion feature learning for the first task. As these aspects are the key to action recognition, we build a novel building block for a 2D CNN-based network for efficient and effective learning of the abovementioned features. Typical previous approaches utilize 3D CNNs to cope with spatial and temporal features while they suffer from massive computations. Other approaches are to utilize (1+2)D CNNs to learn spatial and temporal features efficiently while they need to pay more attention to the importance of motion representations. To overcome problems with previous approaches, we propose a novel block that can capture spatial and temporal features more faithfully and efficiently learn motion features. This proposed block includes Motion Excitation (ME), Multi-view Excitation (MvE), and Densely Connected Temporal Aggregation (DCTA). The purpose of ME is to encode feature-level frames difference; MvE is designed to enrich spatiotemporal features with multiple view representations adaptively; DCTA is to model long-range temporal dependencies. We inject the proposed building block, which we refer to as the META block (or simply "META"), into 2D ResNet-50. We verify the performance of our proposed model on three large-scale benchmarking datasets, including the Something-something v1, Jester, and Moments in Time Mini datasets.

We introduce deep fusion schemes for multi-label multi-class classification for the second task. This approach extends the previous work by introducing a fusion of spatial and temporal branches to provide superior action recognition capability toward multi-label multi-class classification problems. We propose three fusion models with different fusion strategies. We first build several efficient Temporal Gaussian Mixture (TGM) layers to form spatial and temporal branches to learn a set of features. In addition to these branches, we introduce a new deep spatio-temporal branch consisting of a series of TGM layers to learn the features that emerged from the existing branches. Each branch produces a temporal-aware feature that assists the model in understanding the underlying action in a video. We verify the performance of our proposed models on the well-known MultiTHUMOS benchmarking dataset.

Extensive experimental studies prove that both approaches demonstrate competitive results on several benchmarking datasets compared to the baseline and state-of-the-art methods.

Table of contents

List of figures **v**

List of tables **xi**

1 Introduction **1**

1.1 Background 1
1.2 Research Challenges in Video Action Recognition 2
1.3 Research Questions and Focus 4
1.4 Contributions 6
1.5 Thesis Organization 8

2 Background Concepts **11**

2.1 Video Action Recognition 11
2.2 Machine Learning for Action Recognition 13
2.3 Deep Neural Network Fundamentals 19
2.4 Application of Action Recognition 35
2.4.1 Criminal Detection in Video Surveillance 35
2.4.2 Video Recommendation System 36

3 Spatio-Temporal and Long-Term Sequence Features Learning **39**

3.1 Introduction 39
3.2 Related Work 40
3.2.1 Spatial and Temporal Learning 40
3.2.2 Long-Term Sequence Learning 41
3.2.3 Multi-Stream Architecture 43
3.3 Proposed Multi-Stream with Composite of I3D and LSTM 43
3.3.1 Convolution Block with I3D 44
3.3.2 Temporal Block with LSTM 45
3.4 Experiments and Evaluation 46

3.4.1 Dataset 46
3.4.2 Evaluation Measure 46
3.4.3 Implementation 47
3.4.4 Result & Analysis 49
3.4.5 Comparison with Related Work 50
3.4.6 Discussion 50
3.5 Summary 53

4 Motion and Multi-view Excitation With Temporal Aggregation 55
4.1 Introduction 55
4.2 Related Work 57
4.2.1 Switching from 3D to 2D CNN 57
4.2.2 Squeeze and Excitation Mechanism 59
4.2.3 Motion Modeling 60
4.2.4 Temporal Shifting Mechanism 61
4.2.5 Vision Transformer 62
4.3 Proposed Excitation and Aggregation Networks 63
4.3.1 Motion Excitation (ME) 63
4.3.2 Densely Connected Temporal Aggregation (DCTA) 67
4.3.3 Multi-view Excitation (MvE) 69
4.3.4 META Block 73
4.3.5 Discussion with TEA 73
4.4 Experiments and Evaluation 73
4.4.1 Dataset 73
4.4.2 Experimental Setup 75
4.4.3 Evaluation Measure 75
4.4.4 Result & Analysis 76
4.4.5 Comparison with Related Work 77
4.4.6 Discussion 81
4.5 Summary 83

5 Deep Fusion Schemes of Temporal Gaussian Mixture 85
5.1 Introduction 85
5.2 Related Work 86
5.2.1 Three-Stream CNNs 86
5.2.2 Feature Fusion Strategies 87
5.2.3 Temporal Gaussian Mixture 89

5.3 Proposed Deep Fusion of Temporal Gaussian Mixture for Multi-Label Multi-Class Problem 90
5.3.1 Spatial and Temporal Fusion 91
5.3.2 Early Spatio-Temporal Fusion 92
5.3.3 Multi-Level Spatio-Temporal Fusion 92
5.4 Experiments and Evaluation 93
5.4.1 Dataset 94
5.4.2 Experimental Setup 94
5.4.3 Evaluation Measure 95
5.4.4 Result & Analysis 96
5.4.5 Comparison with Related Work 97
5.4.6 Discussion 98
5.5 Summary 100

6 Summary and Future Work **103**
6.1 Summary 103
6.2 Future Work 105

References **109**

List of figures

1.1 Overview diagram of our proposed video action recognition framework. 4

2.1 An example of the data captured by the HOG, HOF, and MBH descriptors. The subject is walking away from the camera as the camera pans from right to left. Color (hue) and saturation indicate the orientation and size of a gradient or flow. The background constantly moves in the optical flow (top, middle), resulting from the camera motions. The relative motion between the subject and the background is encoded by the motion boundaries (right). Image is taken from [81]. 12

2.2 An overview of action recognition framework utilizing two algorithms to classify an action. 12

2.3 Taxonomy of ML algorithms. The area in grey is our dissertation's focus. 14

2.4 Schematic representation of a local features pipeline for action recognition. Different stages of the pipeline are separated by the tallest vertical lines, with stage titles shown at the top. Abbreviations, as stated in the illustration, are described in the previous paragraph. 16

2.5 Supervised classification learning involves input images and their corresponding classes and forwards these inputs to machine learning algorithms to learn the correlation between images and their classes. Finally, its output is a set of probabilities of pre-defined classes. 17

2.6 A Venn diagram explaining how deep learning is the core of artificial intelligence, with data science comprising the other half. 19

2.7 A summary of the architecture of a convolutional neural network (CNN) and the data flow. The building blocks of a CNN model are convolution layers, pooling layers (such as max pooling), and fully connected (FC) layers. Using backpropagation, the gradient descent optimization process updates a parameter that can be learned, such as kernels and weights, based on the loss value. The performance of a model under particular kernels and weights is determined using a loss function and forward propagation on a training dataset. 20
2.9 A convolution process illustrates input image, feature detector (or kernel), and feature maps (output). 21
2.8 An illustration of the relation of softmax and cross-entropy for multi-class classification problems and sigmoid and binary cross-entropy for multi-label classification problems. GT denotes the ground truth. GT will be taken into account to calculate the loss. The calculation result will be backpropagated to the CNN to update the weights. When the network has been optimized, then the network is ready for inference. Notice that the threshold is utilized in the inference phase to decide if the given probability score for a class is above the pre-defined threshold in multi-label multi-class classification. 22
2.10 An input of 4D image data tensor and three CNN layers with different kernel dimensions. 24
2.11 An LSTM layer with units inside it. 25
2.12 A simplified bi-LSTM 26
2.13 An overview of a dropout mechanism. At every iteration, it randomly selects some nodes and removes them along with all their incoming and outgoing connections, as shown. 26
2.14 Some examples of image augmentations. Notice that the transformed images are similar to the original images, except some parts of transformed images are masked out in order to build the deep learning model more general to the other unseen images. 28
2.15 Re-using ResNet-50 for a different task. First, a random kernel initialization of ResNet-50 is utilized for large-scale ImageNet dataset training. After the model has been optimized for ImageNet, a pre-trained model is used for an action recognition task with a smaller number of classes. 30

2.16 An overview of the feature extraction process. The RGB images are fed into the deep learning model pre-trained on ImageNet and Kinetics and obtain meaningful features estimating the underlying action for the input frames. These features are usually saved as a Numpy array for further classification steps. 31
2.17 A ResNet architecture with 50 layers. 32
2.18 A skip connection in each convolution block. 33
2.19 An architecture of an Inception module with dimension reduction layers. 34
2.20 The inflated Inception-V1 architecture. Image is taken from [11]. 34
2.21 An illustration of a suspicious activity detection system. 35
2.22 An illustration of video-based recommendation system. 36

3.1 A network with bi-LSTM on top of the base network to analyze hidden sequential patterns in both temporal and spatial sequential data. 42
3.2 Two related work of video classification from Simonyan *et al.* [70] and Feichtenhofer *et al.* [18]. **(a)** Two-stream architecture to accommodate the spatial and temporal aspects of a video. The spatial stream allows the network to identify an object related to the underlying action; the temporal stream is designated to learn motion representation in encoded optical flow data. **(b)** Two-stream architecture for video classification from [18]. 44
3.3 Overview of multi-stream video classification architecture, consisting of temporal and convolution blocks. 45
3.4 Detail of the proposed model. 48
3.5 The baseline network. 49
3.6 Some prediction result on MIT validation set. 53

4.1 Comparison of three designs of spatiotemporal feature learning. Assuming k is the kernel with the size of 3: (a) vanilla 3D convolution, kernel convolves on temporal (T) and spatial ($H \times W$) axes simultaneously; (b) (1+2)D convolution, temporal and spatial feature learnings are constructed serially; (c) a multi-view design from [49], multi-view feature learnings occur independently, and the resulting feature maps are aggregated with weighted average α_i to produce multi-viewed feature maps. 59

4.2 Feature recalibration on channel responses by squeezing feature maps across spatial dimensions and performing a self-gating mechanism to excite the embedding. 60
4.3 RGB difference at feature-level. 61
4.4 Comparison between tensor without and with part shift mechanism: (a) the tensor without shift mechanism; (b) the tensor with shift mechanism, only part of the tensor is shifted to give an opportunity of the model to learn temporal dimension. 61
4.5 Each clip is passed through the spatial transformer and results in an encoding vector for each clip with CLS token arranged in appropriate position embedding. This positioned embedding is then passed through a temporal transformer encoder. Each token is a vector extracted from a clip for the temporal transformer, so each token is from a different temporal index in the video. 62
4.6 An overview of our proposed model implemented in 2D ResNet50 [31] architecture. We replace the original "conv2" with ME, MvE, and DCTA inside every residual block to construct the META block. Inside Res2Net [21] module, we insert a DCTA submodule. Details on data flow are given in Section 4.4.2. 64
4.7 Two adjacent feature maps are subtracted to obtain motion representation. We firstly apply channel-wise 3 × 3 convolution on frames $[t + 1]$ before subtraction. 65
4.8 An illustration of zero padding. We can devide this illustration into three columns. The first column is the input feature maps. The second column is the splitted feature maps followed by subtraction process. The third column is the motion representation at timestep t. We add zeros to the last timestep. 66
4.9 A detailed diagram showing the architecture of the DCTA submodule inside Res2Net module. 68
4.10 Visualization of decomposed spatiotemporal images in different views. T, H, and W denote temporal size, height, and width, respectively. . . 70
4.11 Detailed architecture of MvE submodule. 71
4.12 Examples of frames from (top-down) Something-something v1 ("*Throwing something*"), Jester ("*Swiping up*"), and Moments in Time Mini ("*repairing*", "*boxing*") datasets. 74

4.13 Ball chart reporting the top-1 accuracy *vs.* computational complexity (in GFLOPs). The size of each ball indicates model complexity. "A & M" corresponds to ACTION-NET [88] and MEST [98], while "S & S" denotes STM [41] and SMNet [94], respectively. We merge their icon since they share similar number of accuracy and GFLOPs. 79
4.14 Examples of prediction results showing three frames and its probability scores (left-right): Something-something v1 ("*Lifting something with something on it*"), Jester ("*Pulling hand in*"), and Moments in Time Mini ("*Tying*") datasets. 81
4.15 Visualization of GradCAM using several methods. 82
4.16 Subfigures a, b, and c illustrate values of MvE coefficients for the Something-something v1, Jester, and Moment in Time Mini datasets, respectively. 83

5.1 The three-stream network's overview at multiple levels (MLTSN). Each of the MLTSN's three streams—spatial, temporal, and multi-level correlation streams (MLCS)—can produce high-level features for classification. 86
5.2 Three streams convolutional neural network (TSCNN) is formulated by fusing scores from Pose ConvNET with video-based scores from spatial and temporal streams. Image is taken from [44]. 87
5.3 A SlowFast network consists of two pathways: a Slow pathway has a low frame rate and low temporal resolution whereas a Fast pathway has $\alpha\times$ higher temporal rate with $\beta\times$ lower channel capacity. Image is taken from [17]. 88
5.4 Inside the TGM layer. A Gaussian weighted kernel with length L is multiplied by each input channel C_{in} to produce a tensor with the shape of $C_{in} \times D \times T$. The figure illustrated above is taken from [60]. 89
5.5 A TGM kernel with multiple temporal convolutions C and length L is computed from learnable attentive parameters and multiple Gaussian distributions. M denotes the number of Gaussian distributions. There are two constrained variables for each Gaussian distribution: a center μ and a width σ. The image is taken from [60]. 90

5.6 The output from 3DCNN is concatenated with the output from the last TGM layer. This shortcut connection prevents information loss during the learning phase. Squeeze and permutation are necessary for the output of base 3DCNN ($T \times 1 \times 1 \times D \rightarrow D \times T$) prior to the concatenation. This figure illustrates only one branch, and thus no fusion occurred. 91
5.7 Detailed architecture of fusion of spatial and temporal streams. 92
5.8 Detailed architecture of fusion of spatial and temporal streams. 93
5.9 Detailed architecture of fusion of spatial and temporal streams. 93
5.10 Some videos of the MultiTHUMOS dataset. Each video may contain multiple activities; hence, the task is categorized as a multi-label multi-class classification problem. The image is taken from [95]. 94
5.11 An example of various activities in a video of a volleyball match. There are five temporal regions to illustrate the lifespan of an activity. Temporal regions in black and blue show the ground truth (label) and the baseline model, respectively. The other colors show the proposed models. We show video images that describe the two actions, "*Run*" and "*VolleyballSpiking*". A red dot indicator is placed below a person performing a specific activity at time t. It is observed that interpolating semantic information between two branches improves performance. Best viewed in color. 97
5.12 An illustration of switching from I3D to META. 100

List of tables

3.1 F1 score of classes showing Top-5 best and Top-12 worst. 50
3.2 Performance of each proposed model on a validation set of Moments in Time Mini dataset 51
3.3 Comparison of model performance with other methods on a validation set. 52

4.1 The comparison result of an individual component, including FLOPs and the number of parameters. We use TSM as a baseline. 76
4.2 Ablative experiments of the number and location of META to verify the contribution of META to the performance. The more META implemented, the more accuracy can be achieved. 77
4.3 The comparison result of META against other state-of-the-art methods on the Something-something v1 dataset. RN in column backbone indicates ResNet. We list UniFormer methods with 16 input frames for relevant comparison. The highest accuracies for CNN-based networks are highlighted in bold. 78
4.4 The comparison result of META against other state-of-the-art methods on the Jester dataset. The listed methods used ResNet-50 as the backbone, except for TRN Multiscale used BNInception. Scoring was done using a validation set. 80
4.5 The comparison result of our work against other CNNs on the Moments in Time Mini validation set. 81

5.1 Comparison with other state-of-the-art methods on a popular benchmark MultiTHUMOS dataset. Even with fewer output channels (C_{out}) for each TGM layer, we consistently outperform the baseline and other methods. 98

5.2 Results for different channel combinations. Interestingly, different modalities yield different results for the same operation. This table also emphasizes that the RGB images are a more critical modality than the optical flow in terms of accuracy. Note that "3" in 3TGM refers to the number of TGM layers used for this ablation experiment. 99
5.3 Results for different weights for each branch. Proposed #3 is used to produce this result. 99

Chapter 1

Introduction

1.1 Background

One of the critical issues with computer vision is video interpretation. Videos offer a temporal component via which motion and extra information can further the image recognition task's informational capacity. One of the intricate parts of computer vision that attracts many academics to investigate is action recognition, which is the initial stage in video action recognition. Video provides temporal information that creates motion instead of image recognition issues. Background clutter, partial occlusion, noise, shifting camera angles, and other issues might be present in the video. These challenging circumstances motivate researchers to use novel methods for improving accuracy.

Meanwhile, there are various benefits to the development of information technology, including its applications and infrastructure. Due to the fast expansion of cloud and edge computing, we are used to using social media and being in public view. While this is happening, several companies, particularly those in the security and transportation sectors, gather a significant library of videos packed with knowledge on human behavior, traffic patterns, and other topics. More scholars are growing interested in video interpretation as the volume of video data rises. Action recognition, which seeks to identify human actions in films, is the first stage of the video interpretation process. The spatiotemporal and mobility variables are the essential components for action recognition.

Until convolutional neural networks had significant success, it was believed that video action recognition research successfully stored spatio-temporal patterns defining local properties from videos using traditional methodologies. Motion Energy Image (MEI) and Motion History Image are two examples of these traditional methods for depicting activity (MHI). These traditional methods were considered pioneering efforts to identify human action. Traditional methods like the Histogram of Oriented Gradients (HOG), Histogram of Optical Flow (HOF), and Motion Boundary Histogram are also utilized to derive trajectories from the movies (MBH). The local descriptor encoded using one of these techniques is then projected as a global feature using many feature processing techniques, such as Bag-of-Word (BoW), VLAD, or Fisher vector encodings. By merging spatio-temporal local data to create global video representations, these approaches can produce state-of-the-art results in various video recognition benchmarks.

Deep convolutional neural networks (CNN) have shown exceptional performance in large-scale picture classification. Because they learn a hierarchy of feature representations through end-to-end optimization, CNNs perform better than traditional hand-crafted features. It works remarkably well for other related tasks, including object detection, semantic segmentation, and picture retrieval. Several network enhancement ideas have been put forth, including AlexNet, VGG, Inception, and ResNet. 2D CNN-based frameworks and 3D CNN-based frameworks are the most popular approaches now in use. Spatiotemporal modeling has proven 3D CNNs to be more successful than naive 2D CNN-based models, which can only capture spatial dimensions. Utilizing the two-stream design is another strategy. It was suggested to incorporate optical flow modalities and spatiotemporal information, which significantly improved performance compared to the one-stream design.

Applying deep networks for video interpretation has come with many difficulties. To understand temporal and spatial appearance, suitable models must first be created. To fully comprehend videos, both short-term motion and long-term context are needed. Second, processing videos requires more computing since they have one more dimension than images. Large volumes of data require the processing of practical algorithms.

This dissertation investigates the challenge of using deep networks for video classification tasks. We present different methods for recognizing actions in videos that concentrate on gathering long-term temporal data and capturing both local and short-term motion.

1.2 Research Challenges in Video Action Recognition

This dissertation focuses on the action recognition problem in videos recorded under various environmental conditions varying from a constant, clean background to complex, cluttered, and moving backgrounds. This dissertation aims to address current action recognition deficiencies by introducing novel feature extraction, representation, and classification methods to improve performance. Several challenges need to be addressed to improve the model in predicting pre-defined label(s), including differentiating symmetric action pairs, building a lightweight model, estimating motion representations, learning spatial and temporal reasoning, handling long-range temporal evolution, predicting labels in multi-label classification problems, and localizing temporal action.

- **Symmetric action pairs.** The complex actions in videos are normally temporal-dependent, containing spatial information within each frame and including temporal information over a duration. For example, symmetric action pairs (*'opening the door'*, *'closing the door'*), (*'plug USB cable'*, *'unplug USB cable'*) contain similar features in spatial domains, but the temporal information is completely reversed.

- **Lightweight model for real-time application.** A deep learning model that is lightweight and fast inferences is more favor rather than heavy and slow when inferencing a video.

- **Frame difference calculation.** Utilizing optical flow modality to emphasize the motion will boost the performance but is unsuitable for real-time application. Thus, an alternative method for calculating the frame difference is important.

- **Spatial and temporal reasonings.** A benchmarking video for action recognition may contain spatial and temporal information that needs to be processed to classify an action. Another video may contain only spatial or temporal aspects to consider when classifying an action. The model thus needs to be generalized for all types of video.

- **Long-range temporal relationship.** Temporal modeling is a key for video action recognition. It usually considers both short-range motions and long-range aggregations. It is vital to design a neural network that effectively captures long-range temporal evolution, minimizes temporal information loss, and makes network optimization easy.

- **Multi label classification problem.** Temporal action segmentation consists of labeling each video frame with more labels from a list of action categories rather than one label. The task is sometimes called "frame-wise action classification". Example datasets include Charades [69] and MultiThumos [95].

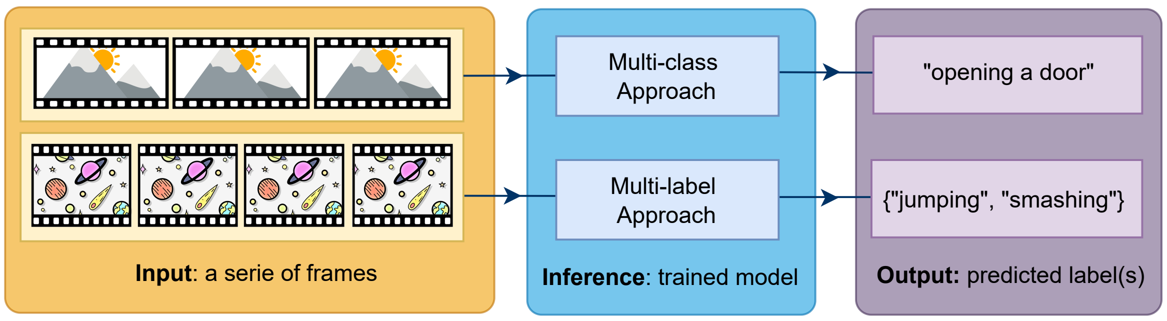


Fig. 1.1 Overview diagram of our proposed video action recognition framework.

- **Temporal action localization.** Temporal action detection aims to determine the semantic label and the temporal interval of every action instance in an untrimmed video. The task of action localization in the untrimmed video involves predicting the action, or actions, present at each time step of the video sequence. It is a fundamental and challenging task in video understanding.

1.3 Research Questions and Focus

This thesis focuses on building a deep learning-based model to translate the spatial and temporal representation into a predefined action. This research task is known as the action recognition task (Figure 1.1), with the specific input being a sequence of frames, and the output is predefined action label(s). This research employs several datasets for benchmarking purposes, each with a different characterization. Generally speaking, this research contains two types of approach according to the dataset type: (1) multi-class classification approach and (2) multi-label multi-class approach.

In the first approach, we propose a lightweight spatiotemporal feature learning, including excitation and aggregation mechanisms. The proposed approach aims at better temporal modeling and is lightweight compared to 2D/3D CNNs counterparts. The excitation submodule attempts to estimate the motion representations at the feature level and represent feature maps from different views. In contrast, the aggregation mechanism is to capture local temporal events in a hierarchical structure with residual connections between every adjacent subset. The second approach is designed to overcome a laborious computation and storage space demanding optical flow modality. It also focuses on optimization difficulty when local convolutions are repeatedly stacked. Moreover, we alleviate such a heavy computation of 3D convolution by (i) factoring the 3D kernel into (1+2)D kernel configuration and (ii) splitting an input view into multi-view shapes that fits a 2D kernel.

The second approach is multi-stream with the temporal gaussian mixture. Unlike common classification tasks where class labels are mutually exclusive, multi-label classification requires algorithms that support predicting multiple mutually non-exclusive classes or labels. A remaining unexplored problem for the multi-label classification task is the absence of fusion strategies between branches. Therefore, we exploit the latent correlation between the RGB and optical flow streams and produce a new stream of mingled feature maps for multi-label multi-class classification problems. We perform several experiments for multi-label multi-class problems and investigate several fusion methods to achieve competitive results on a benchmarking dataset. We also explore the possibility of channel combination of a temporal gaussian mixture layer on both RGB and optical flow streams to maximize the result.

The list of research questions for the two suggested techniques that this dissertation is supposed to address is presented in this section. Brief research questions related to the proposed approach of excitation and aggregation mechanisms (first approach) are summed up as follows:

- Q1.1. Do motion and multi-view excitation followed by a spatiotemporal aggregation approach improve the deep learning action recognition model?
- Q1.2. How can we tackle the optical flow modality problem for action recognition?
- Q1.3. How to emphasize the important view adaptively in multi-view configuration?
- Q1.4. How do we build and integrate excitation mechanisms of motion and multi-view approach in a unified deep learning model?
- Q1.5. How to achieve long-range temporal relationship modeling for action recognition?
- Q1.6. How successful is our proposed method of predicting an action in a video compared to the baseline and various state-of-the-art methods?

Brief research questions related to the proposed approach of multi-stream temporal gaussian mixture strategy (second approach) are summed up as follows:

- Q2.1. Can a fusion scheme be effective for improving the model performance in multi-label multi-class classification problems?
- Q2.2. How to extract features for both RGB and optical flow streams for further temporal learning?

Q2.3. How can we formulate effective fusion strategies for RGB and optical flow data?

Q2.4. What kind of channel-wise combination strategies are effective for each temporal gaussian stream?

Q2.5. Our proposed fusion schemes consist of RGB and optical flow streams. How do we implement two streams effectively, considering each branch may contribute inequable performance?

Q2.6. How successful is the proposed multi-stream temporal gaussian mixture approach compared to the baseline and other state-of-the-art methods on the same benchmarking dataset?

1.4 Contributions

This dissertation focuses on classifying an action in a video into a predefined label. We propose several novel approaches containing some contributions to improve and optimize the model's performance. This research's key contributions will be summarized and linked to the related research questions; thus, it will help clarify the problem mapping to the contribution solution. We will also mention the section parts where the discussion of the question and contribution occurs to make it easy to point out. In this dissertation, two main contributions for action recognition can be categorized: (1) the contribution of the proposed strategy to improve the performance by adding excitation and aggregation mechanisms into convolutional-type of deep learning model; (2) the contribution of fusion schemes in temporal-gaussian-mixture based multi-stream architecture.

C.1. The contribution of motion and multi-view excitations together with densely temporal aggregation mechanism for action recognition in response to short-range and long-range temporal evolution problems. The more detailed contributions related to the main contribution are listed below:
[*The contributions are related to Q1.1., which will be discussed in Section 4.3*]

C1.1 Motion modeling is one of the key concepts to video understanding tasks. Motion modeling involves calculating every adjacent frame and producing a single image showing pixel displacement over time. Optical flow is a common technique for calculating short-range motion estimation. Due

to its high computational demands and storage space requirements, this approach is inappropriate for real-time applications. Several researchers have begun integrating motion modeling into a unified deep neural network. In our work, we integrate a motion modeling technique into a 2D CNN and inject an excitation mechanism into that technique to force the network to focus on an important part of input channels.
[*The contributions are related to Q1.2. and Q1.4., which will be discussed in Section 4.3.1*]

C1.2 In the multi-view submodule, three-dimensional input feature maps are split into three branches, and each branch will perform two-dimensional kernel learning. The resulting feature maps are further aggregated in our configuration with a weighted average. By design, these weights are end-to-end optimized throughout the training phase to guarantee that weight adjustment is carried out flexibly.
[*The contributions are related to Q1.3. and Q1.4., which will be discussed in Section 4.3.3*]

C.1.3 On the action recognition task, temporal modeling is another key concept that the network needs to focus on. Our proposed submodule of temporal modeling consists of several shared local temporal convolutions stacked up and densely connected. Local temporal convolution is targeted to learn long-range temporal relations. We arrange this submodule inside the Res2Net module. The Res2Net module is composed of four subgroups of convolutions, i.e., input features are divided into four branches or subgroups.
[*The contributions are related to Q1.5., which will be discussed in Section 4.3.2*]

C.2. We propose several fusion schemes of RGB and optical flow streams for multi-label multi-class classification problems. Specifically, we propose three approaches to fuse the streams for accurate results: (i) spatial and temporal fusion, (ii) early spatio-temporal fusion, and (iii) multi-level spatio-temporal fusion. Other contributions are listed below:
[*The contributions are related to Q2.1. and Q2.3., which will be discussed in Section 5.3*]

C2.1 Each modality's characteristics must be extracted, i.e., RGB and optical flow, before performing long-range temporal learning utilizing a gaussian

mixture layer. Choosing a deep learning model for that task is important since we expect to have strong action representations encoded in feature vectors. We also need to consider which layer to serve as an endpoint for feature extraction since a deep learning model may consist of many layers.
[*The contributions are related to Q2.2., which will be discussed in Section 5.4*]

C2.2 Inside the temporal gaussian mixture layer, we modify the original temporal gaussian mixture to have less computation when generating output channels. The author implemented a convolution kernel to yield output channels in the original work. Our modification replaces that convolution kernel with a simple function aggregating all input channels into one output channel.
[*The contributions are related to Q2.4., which will be discussed in Section 5.4.6 (Custom Channel Unification)*]

C2.3 We propose several deep learning models which take two modalities as input images. We also design the network having deep fusion schemes. From many works of literature, which will be presented in the latter chapters, we understand that each modality contributes differently to the model's performance. Therefore, having a weighted stream scenario in our proposed model is common.
[*The contributions are related to Q2.5., which will be discussed in Section 5.4.6 (Weighted Branch)*]

Other contributions of this research include a thorough review of the literature in the action recognition domain and related improvements of our proposed approaches over the state-of-the-art methods.
[*The contributions are related to Q1.6. and Q2.6., which will be discussed in Section 4.4.5 and 5.4.5, respectively*]

1.5 Thesis Organization

The remainder of this thesis is organized as follows:

- In **Chapter 2**, we provide some relevant background theories and fundamental concepts underlying this dissertation. The related background concept includes video action recognition, machine learning for action recognition, deep neural network-based solution, and application of action recognition.

- In **Chapter 3**, a preliminary stage of this research, we explain the proposed approach's details, including the multi-stream network approach to accommodate RGB and optical flow modalities.
- In **Chapter 4**, we focus on building a novel architecture to overcome the limitations stated in **Chapter 3**. It includes detailed problems, proposed strategies, and experiment scenarios to explain the effectiveness compared to the other related research.
- In **Chapter 5**, we focus on several fusion strategies in the multi-stream network for multi-label multi-class classification problems. We present three fusion schemes to fuse spatial and temporal features to enhance accuracy. This approach is designed to solve the limitation discussed in the previous chapter.
- In **Chapter 6**, we are summing up the overall results of this dissertation and identifying the remaining challenges for future works.

Chapter 2

Background Concepts

2.1 Video Action Recognition

One of the critical issues with computer vision is video interpretation. Videos add a temporal component via which motion and additional information can be employed to further the image recognition task's informational capacity. This dissertation extends deep convolutional networks to video understanding by modeling spatial and temporal information. The use of convolutional networks is motivated by the success of this method in image classification.

Extracting local features from videos that encode local spatio-temporal patterns has historically proven successful in video action recognition studies. Hand-crafted feature descriptors such as Histogram of Oriented Gradients (HOG), Histogram of Optical Flow (HOF), Motion Boundary Histogram (MBH), and trajectories are utilized to distill useful information from the videos. Figure 2.1 shows a visual comparison between these descriptors. These local descriptors are then encoded to produce a global video-level feature representation with Bag-of-Word (BoW), VLAD, or Fisher vector encodings. These techniques can achieve state-of-the-art results in various video recognition benchmarks by aggregating spatio-temporal local data to produce global video representations.

In large-scale image classification, deep convolutional networks have demonstrated outstanding success. CNN's outperform conventional handmade features because they learn a hierarchy of feature representations through end-to-end optimization. It performs exceptionally well in other related tasks like object detection, semantic segmentation, and picture retrieval. Various network architectures for improvement, including Inception and ResNet, have been proposed. Applying deep networks for video interpretation has come with several difficulties. To understand temporal and spatial

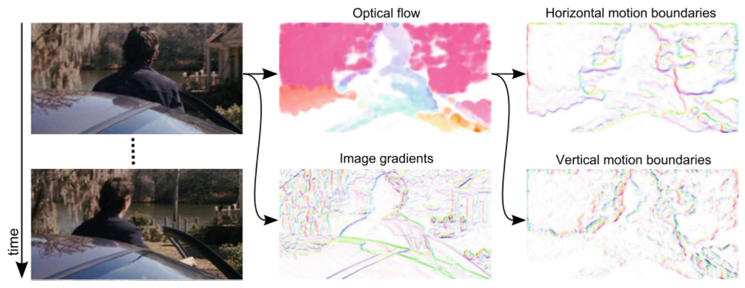


Fig. 2.1 An example of the data captured by the HOG, HOF, and MBH descriptors. The subject is walking away from the camera as the camera pans from right to left. Color (hue) and saturation indicate the orientation and size of a gradient or flow. The background constantly moves in the optical flow (top, middle), resulting from the camera motions. The relative motion between the subject and the background is encoded by the motion boundaries (right). Image is taken from [81].

appearance, suitable models must first be created. To fully comprehend videos, both short-term motion and long-term context are needed. Second, processing videos require more processing since they have one more dimension than images. Large volumes of data require the processing of practical algorithms. An illustration of the action recognition framework is described in Figure 2.2.

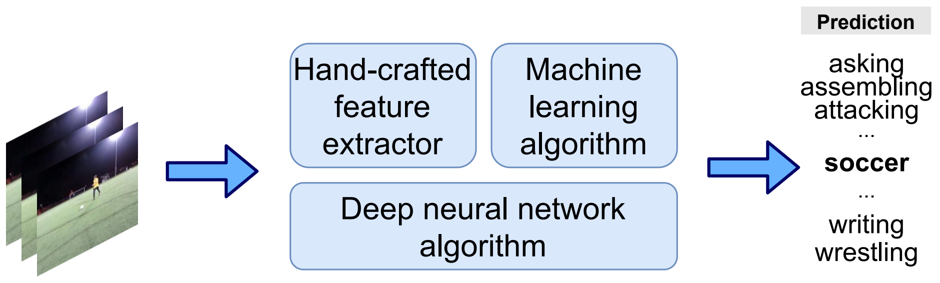

Hand-crafted feature extractor
Machine learning algorithm
Deep neural network algorithm
Prediction
asking
assembling
attacking
...
soccer
...
writing
wrestling

Fig. 2.2 An overview of action recognition framework utilizing two algorithms to classify an action.

Due to its multi-modal (video, audio, subtitles, and commentary) character and the intricacy of the video collecting process, it is challenging to interpret the video. These techniques are helpful at times. In the professional broadcasting of sports videos, complementing signals to visual elements, such as audio, could be very helpful. Along with the video, the movie's script may aid our comprehension of its subject matter. Most of the time, these complementary signals, like music playing in a sports film, are either unavailable or need to be clarified. Only the visual content understanding of videos will be addressed in the scope of this dissertation. Understanding videos involves many issues. Here, we list the research community's most prevalent issues.

- Video understanding
- Action recognition
- Object tracking
- Optical flow computation
- Video content description
- Semantic video segmentation

Although not comprehensive, the list above offers us an idea of the various tasks involved in video interpretation. These issues present a researcher with a variety of difficulties. For instance, object tracking, optical flow estimates, and video compression require keeping track of moving parts. The description of video material and comprehension of action require content identification.

In this dissertation, we study the problem of video classification with deep networks. We present multiple approaches for video action recognition that focus on aggregating long-term temporal information and capturing short-term motion and local appearance.

2.2 Machine Learning for Action Recognition

Many attempts have been made in action recognition to convert action videos into discriminative and representative features to minimize with-in-class variations and maximize between-class variations. Here, we concentrate on hand-crafted action depiction techniques, whose parameters have already been decided upon by specialists. Deep networks, which can automatically learn parameters from data, are different from this.

Machine learning (ML) is a study of algorithms and statistical models, and it is a subset of artificial intelligence. Instead of relying on patterns and inference, computer systems use machine learning (ML) to complete jobs without specific instructions. ML is thus applicable to software engineering, pattern recognition, and computer vision. Because ML uses data to make judgments and requires little assistance from software programmers, it may be applied in an exciting way across a wide range of sectors. Based on the desired output of the algorithm, ML algorithms are categorized into a taxonomy (see Figure 2.3). Standard algorithm types can be classified initially as

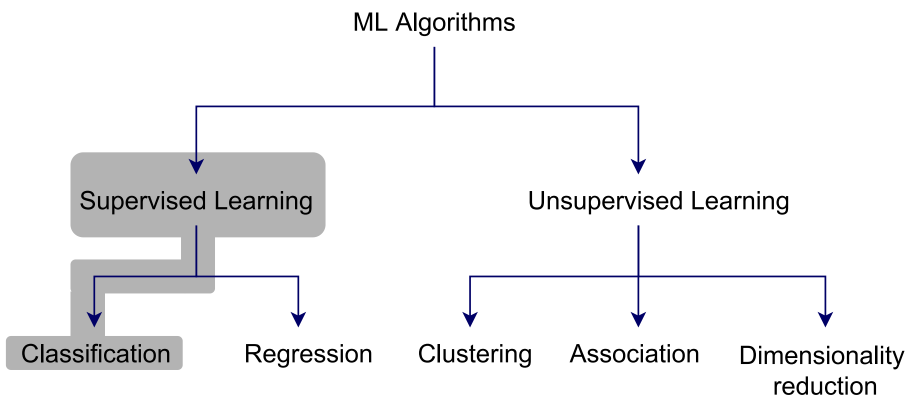


Fig. 2.3 Taxonomy of ML algorithms. The area in grey is our dissertation's focus.

supervised learning and unsupervised learning. The ML algorithms are applied to the training datasets. When a new data instance is used for the classification label prediction, the machine learning algorithm operates on the new instance and determines its classification using historical data and records. Numerous ML algorithms have been created, crafted, and modified to examine action and activity learning. Nearly as many different ML model types have been used for action and activity learning as different sensors are used to collect the data. Regrettably, the hand-crafted features-based encoding methods for video-based action recognition are shown as universal visual and do not consider much temporal information.

Figure 2.4 illustrates the general structure of a pipeline for local feature encoding for dense trajectories. In the subsequent paragraph, we present a short discussion for each stage.

Video processing. In this stage, frames are extracted from a video. The extraction may contain frame skipping, i.e., extract only 10 frames per second (FPS) from a 30FPS video. After that, the potential step may involve data augmentation, which will be discussed more in the upcoming section.

Local feature extraction. The second pipeline stage aims to apply a local feature extractor to the preprocessed frames to extract various local features from them. Examples of methods include the sequence of pixel displacements within the pixel trajectory (Trajectory), Histograms of Gradients (HOG), Histograms of Optical Flow (HOF), Motion Boundary Histograms (MBH), and Scale Invariant Feature Transform (SIFT). Hand-crafted features like SIFT descriptors, typically encoded into bag-of-words (BoW) histograms, are the foundation of traditional methods for image retrieval. An advanced work, RootSIFT from [3], was offered as a solution to the burstiness issue, enhancing the discriminative power of SIFT descriptors by applying the Hellinger kernel-based distance to the original SIFT descriptors. The resulting local feature descriptors are then subjected to PCA and whitening before encoding to reduce the number of dimensions in the descriptors.

Feature encoding. The pipeline's third stage encodes the local features that were previously extracted and post-processed into intermediate feature-level representations. Several encoding techniques are Fisher Vector [6] and Vector of Locally Aggregated Descriptors (VLAD) [38]. VLAD, proposed by Jegou *et al.*, is proposed to obtain a compact representation as a replacement for BoW histograms, which achieves good results while requiring less storage. Another work by Li *et al.* [51] introduced Multi-VLAD, based on constructing and matching VLAD features of multiple levels from an image, to improve localization accuracy. Other global features such as GIST descriptors and Fisher Vector (FV) have also been evaluated for large-scale image retrieval [59, 79, 90]. All of the encoding methods mentioned above are founded on the same fundamental idea: building a common vocabulary of local features that represents the most prevalent features typically present in the input videos, then encoding all local features based on how closely related to vocabulary elements the extracted local features are.

Representation extraction. The variable number of local encodings created in the previous stage are combined into a single, fixed-length vector representing the full video in the fourth stage of the pipeline. Pooling is one method of combining these local encodings. Three pooling strategies include max-pooling, sum-pooling, and average pooling.

Classification. The fifth and final stage of the pipeline's workflow involves taking each video's encoded contents as a single, fixed-size vector and classifying it according to one or more pre-defined classes. Examples of possible classifiers include Neural Networks (NN), Support Vector Machines (SVMs), or Random Forests (RFs). However, due to the possible high dimensionality of the video representations that originated in the previous stage, some classifiers may be more efficient than others. Most works in the literature have successfully used the SVM for this purpose [83, 85, 82], with its linear kernel version being shown to be highly appropriate for large-scale multi-class and multi-label learning.

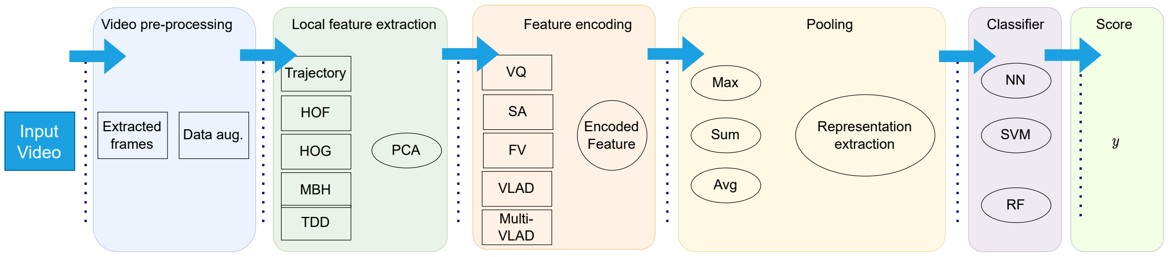


Fig. 2.4 Schematic representation of a local features pipeline for action recognition. Different stages of the pipeline are separated by the tallest vertical lines, with stage titles shown at the top. Abbreviations, as stated in the illustration, are described in the previous paragraph.

Supervised Learning

Many real-world issues can be mathematically formulated for computing by mapping the input space X into the output space Y. As an illustration, in the classification issue, a black box computing model should accurately identify the probability class of Y as the output given an array of inputs X due to the recorded observation. A black box refers to a function whose mathematical formulation is challenging to write down and whose input-to-output function mapping solution is still uncertain. The machine learning algorithm can be split into supervised and unsupervised learning based on the learning method. Both supervised and unsupervised learning requires the representation of observation data for the learning model; the difference is in the availability of the target class category on each observation data for unsupervised learning, which lacks a class label for model training.

Supervised learning algorithms are machine learning algorithms that are learned to map input data into a target attribute of the data (class), as seen in Figure 2.5. The algorithm typically learns a relationship between the properties (features) of the input data and the target attribute by minimizing a loss function specified on the pairs of input data and the matching destination attribute. The discovered relation is represented in a structure referred to as a model. Models explain the unobserved relationships between the input data and the desired attributes and can be used to predict the objective attribute (label), given the input data value.

Building a model of the distribution of class labels in terms of input features is the goal of supervised learning. When the input feature is known, but the class label value is unknown, the resulting classifier gives class labels to the test examples. The most prevalent type of machine learning algorithm used in solving real-world issues is supervised learning, often known as supervision machine learning, whether

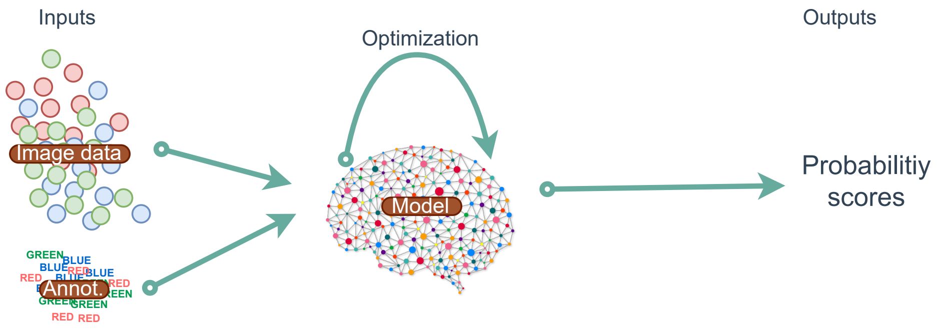


Fig. 2.5 Supervised classification learning involves input images and their corresponding classes and forwards these inputs to machine learning algorithms to learn the correlation between images and their classes. Finally, its output is a set of probabilities of pre-defined classes.

deep or shallow. We will discuss the dataset and its component before diving into further depth regarding supervised learning. A dataset is a collection of data points or records obtained during the training process from actual world observations. A data point is typically represented by one or more properties, also known as data features. Sometimes, the target class's unique attribute, which describes the supervised learning dataset, is added to the dataset. A supervised learning method, in particular, aims to link input features X to the specified target class Y. The ground truth comes through observation and is referred to as the dataset's target class or the desired outcome. Examples of datasets for supervised learning-based action recognition tasks are the Something-something v1, Jester, and Moments In Time Mini datasets.

Classification falls under supervised learning, where the input datasets are also made available to the objectives. Predicting the class of a set of data is done through classification. Classes are occasionally referred to as targets or labels. The objective is to find and identify these activities in films using the datasets' various actions. There are two steps in the classification process for data: First, there is the learning phase, where the goal is to develop a classifier by examining the labeled data; next, there is the predicting phase, which is based on the well-known model for forecasting. This model has sufficient generalization ability, which indicates that it performs well for future data that are predicted to have the same statistical distribution as the training data and has excellent classification accuracy on the training data.

Multi-Class Classification

In multi-class classification, some classes are presumptively present, and each sample is given a single label. The challenge of categorizing occurrences into one of three or more classes is known as multi-class classification or multinomial classification (classifying instances into two classes is called binary classification). The multi-class classification assumes that each sample is given one and only one label. For example, a fruit can only be an apple or a pear at any time. The following will serve as representations of the main classification algorithms:

- Support Vector Machine (SVM). The researchers maximize the distance between a hyperplane that distinguishes two types of data from instances on either side of it. They can perform linear and non-linear separation with a kernel function. Furthermore, the global minimum is reached, and a local minimum is prevented, which can occur in other search algorithms, including neural networks. Finally, they usually delivered decent results.

- K-Nearest-Neighbors (KNN). It identifies the closest K-nearest instances to the query instance and decides its label by choosing the single most common label of the nearest instances. The critical drawback of this classifier is that it requires the storage of all the instances, and it is sensitive to comparing instances to the choice of the similarity function. Besides, it is widely accepted that it is subject to irrelevant features.

- Random Forest classifier (RF). It uses several decision trees and gives the class label based on votes from each decision tree. In addition, a random set of features is used to separate each node. The RF can overcome the restriction of decision trees because the training data are always over-fit.

Multi-Label Classification

Multiple nonexclusive labels may be assigned to each occurrence in multi-label or multi-output classification, a form of the classification issue in machine learning. Multi-label classification is an extension of multiclass classification, which is the single-label issue of classifying occurrences into exactly one of more than two classes. In the multi-label problem, the labels are nonexclusive and there is no limit on the number of classes an input can be assigned to.

Certain machine learning algorithms natively enable multi-label classification. Neural networks may directly enable multi-label classification by defining the number of target labels in the issue as the number of nodes in the output layer. For instance, a job with three output labels (classes) will require an output layer of a neural network with three nodes. Each output layer node must employ sigmoid activation. This will forecast a class membership probability between 0 and 1 for the label. The final step is to fit the model using the binary cross-entropy loss function.

To configure a neural network model for multi-label classification, the following details are necessary: (1) number of nodes in the output layer matches the number of labels, (2) sigmoid activation for each node in the output layer, and (3) binary cross-entropy loss function.

2.3 Deep Neural Network Fundamentals

What is Deep Learning?

A branch of machine learning called "deep learning" trains computers to mimic human behavior (see Figure 2.6). Deep learning algorithms, also known as artificial neural networks, are therefore motivated by the organization and operation of the human brain (ANN). An artificial neural network (ANN) comprises a network of connected artificial neurons. The layers of these neurons are arranged. Every neuron in a layer is a mathematical function that receives input, alters it for use in other calculations, and produces an output. End-to-end learning, which refers to the transformation from pixel level to action categorization, is a notion introduced by learning-based representation techniques that can automatically learn the feature from the raw data. This approach contrasts with handcrafted representation-based techniques, where the activity is represented using handcrafted feature detectors and descriptors.

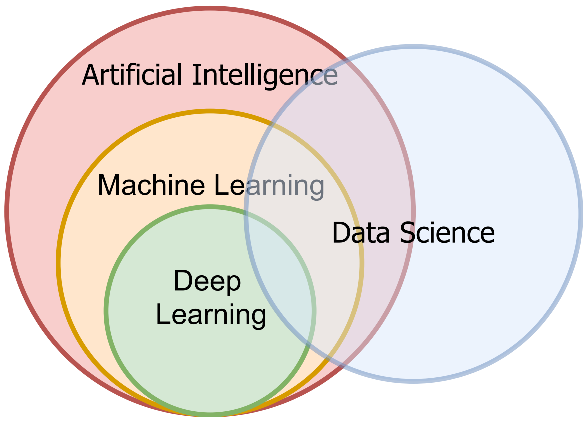


Fig. 2.6 A Venn diagram explaining how deep learning is the core of artificial intelligence, with data science comprising the other half.

In the following paragraph, we will discuss several essential terms or operations that are involved in the world of deep learning.

Forward Propagation and Backward Propagation

When training an artificial neural network, the forward propagation mechanism feeds input x through the layers of the network, producing the output y, as depicted in Figure 2.7. The input x provides information to hidden units at each layer with weighted connections to produce y. The objective function is the other component in the learning process of artificial neural networks that responds to measure the success of the output quality from a model. Information from the objective function must flow in reverse to estimate the gradient of the cost or error in order to update the weighted trainable parameters and bias.

The backpropagation algorithm is an algorithm to compute the gradient by flowing backward of information cost through the network (see Figure 2.7). The backpropagation algorithm is maybe the essential element of a neural network. The program uses a chain rule technique to train a neural network efficiently. After each forward pass, backpropagation makes a backward pass through a network while updating the model's parameters (weights and biases). It resolves the issue that calculating the gradient of a function is computationally costly and inefficient. The essential concept of the backpropagation method is to simulate a given function by modifying the internal weightings of input signals to generate a predicted output signal. The error between the system's output and a known expected output is used to modify the system's internal state during training using a supervised learning technique.

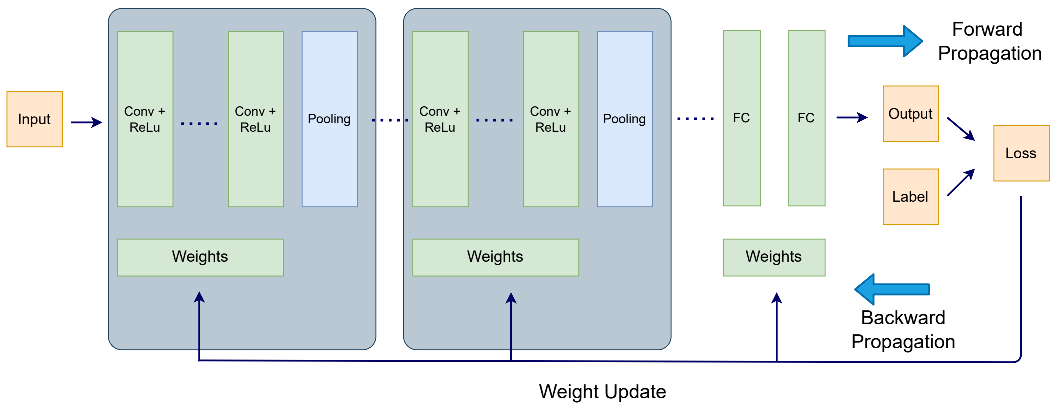


Fig. 2.7 A summary of the architecture of a convolutional neural network (CNN) and the data flow. The building blocks of a CNN model are convolution layers, pooling layers (such as max pooling), and fully connected (FC) layers. Using backpropagation, the gradient descent optimization process updates a parameter that can be learned, such as kernels and weights, based on the loss value. The performance of a model under particular kernels and weights is determined using a loss function and forward propagation on a training dataset.

In a multilayer feed-forward neural network, the backpropagation algorithm is a technique for training the weights. As a result, it necessitates the definition of a network structure consisting of one or more levels, each of which must be fully connected to the previous layer. One input layer, one hidden layer, and one output layer make up a typical network topology.

Both classification and regression issues can benefit from the use of backpropagation. The network performs best in classification challenges when it has one neuron in the output layer for each class value. For instance, a two-class or binary classification issue involves the class values A and B. It would be essential to translate these expected outcomes into binary vectors containing a column for each class value, such as [1, 0] and [0, 1] for A and B, respectively. This encoding method is known as one hot.

As mentioned in the previous section, there are commonly two problems for classification, i.e., multi-class and multi-label classification. The basic diagram for both is illustrated in Figure. 2.8. When a solution is applied for a multi-class classification problem, it can be set up too for multi-label multi-class classification. Specifically, we only switch the activation and loss functions from softmax and cross-entropy to sigmoid and binary cross-entropy, and we obtain multi hot vector, e.g. [1,0,1,1] instead of one hot vector [0,0,0,1].

Convolutional Neural Network

Recent advancements in convolutional neural network (CNN) performance in various action recognition tasks is impressive. CNN employs layers with local feature-applied convolving filters to interpret data with a grid pattern (see Figure 2.9 for illustration).

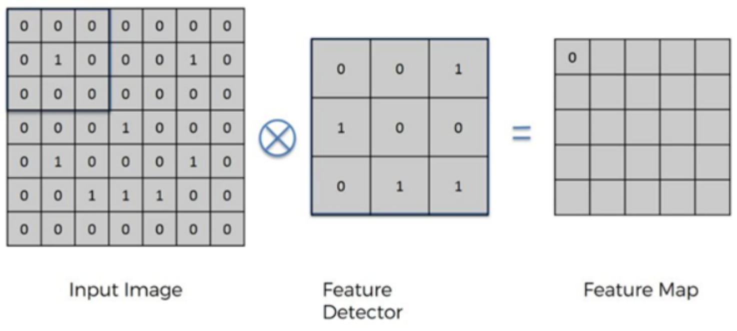


Fig. 2.9 A convolution process illustrates input image, feature detector (or kernel), and feature maps (output).

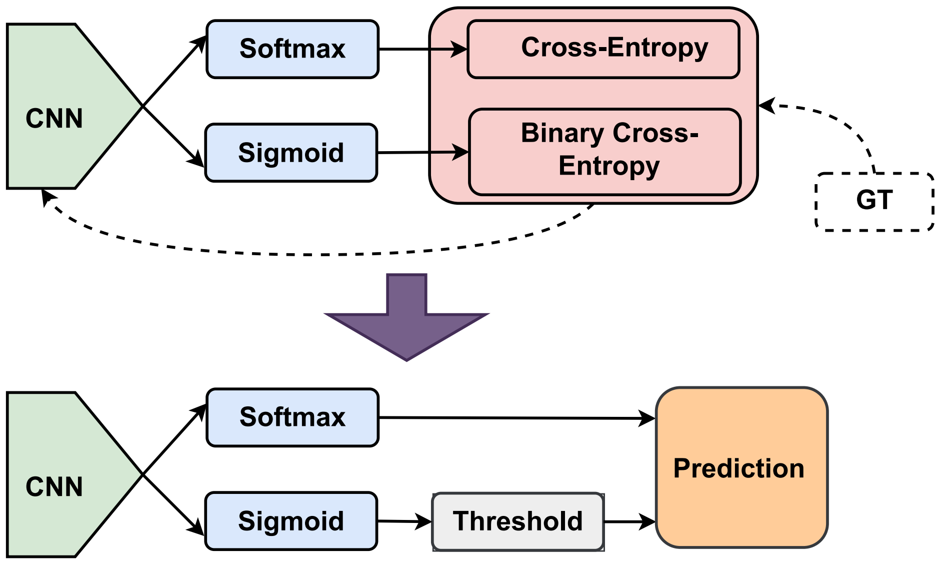


Fig. 2.8 An illustration of the relation of softmax and cross-entropy for multi-class classification problems and sigmoid and binary cross-entropy for multi-label classification problems. GT denotes the ground truth. GT will be taken into account to calculate the loss. The calculation result will be backpropagated to the CNN to update the weights. When the network has been optimized, then the network is ready for inference. Notice that the threshold is utilized in the inference phase to decide if the given probability score for a class is above the pre-defined threshold in multi-label multi-class classification.

CNN was created with the animal visual cortex in mind [35, 19] and is intended to automatically and adaptively learn spatial hierarchies of characteristics, from low- to high-level patterns. Convolution, pooling, and fully connected layers are the three types of layers (or "building blocks") that make up a standard CNN. Convolution and pooling layers in order one and two do feature extraction, whereas a fully connected layer in order three maps the extracted features into the output, such as classification. A convolution layer plays a crucial role in CNN, composed of a stack of mathematical operations, such as convolution, a specialized linear operation.

Since a feature can exist anywhere in a digital image, the pixel values are kept in a two-dimensional (2D) grid. An optimizable feature extractor, a small grid of parameters known as the kernel, is applied at each image point. This process makes CNN exceptionally useful for image processing. Extracted features may gradually and hierarchically become more sophisticated as one layer feeds its output into the following layer. Training minimizes the difference between outputs and ground truth labels using an optimization technique like backpropagation and gradient descent, among others. It involves improving parameters such as kernels.

The CNN architecture includes several building blocks, such as convolution layers, pooling layers, and fully connected layers. A typical architecture consists of repetitions of a stack of several convolution layers and a pooling layer, followed by one or more fully connected layers. The step where input data are transformed into output through these layers is called forward propagation, as illustrated in Figure 2.7.

Next, we will describe a basic CNN model that can be applied to the action recognition of a video. We divide CNNs based on the space it learns, spatiotemporal, spatial, and temporal space learning. The visual difference can be seen in Figure 2.10.

The essence of action recognition is to obtain the appearance information in space and the movement information in time. A single 3D convolution can capture the spatiotemporal information of video action behaviors at the same time. The 3D convolutional neural networks are designed by researchers for action recognition and can directly extract information from multiple video frames through 3D convolution. 3D CNN proves the rationality of 3D convolution to extract spatiotemporal features. Although 3D CNN-based approaches have achieved exciting results on several benchmark datasets, this type of CNN has many parameters and a large amount of calculation. This excessive calculation is mainly due to the increased dimension of 3D convolution. In this case, various problems are caused, such as easy overfitting and difficulty in converging, and pose challenges, including ineffective inferences for online streaming videos in real-world applications.

As 3D convolution leads high computational load, few works aim to reduce the complexity by decomposing the 3D convolution into 2D spatial convolution and 1D temporal convolution. The 2D CNN method performs convolution on one frame at a time without temporal learning. To improve their accuracy, most 2D CNN mainstream approaches, such as two-stream, incorporate optical flow modality into account. However, this leads to additional computational costs when involving the optical flow. Therefore, several researchers studied to include temporal learning for 2D CNNs. This approach is targeted to maintain the low model complexity and computational cost while still posing strong results on several well-known benchmarking datasets; hence the lightweight model can be implemented in real-time applications.

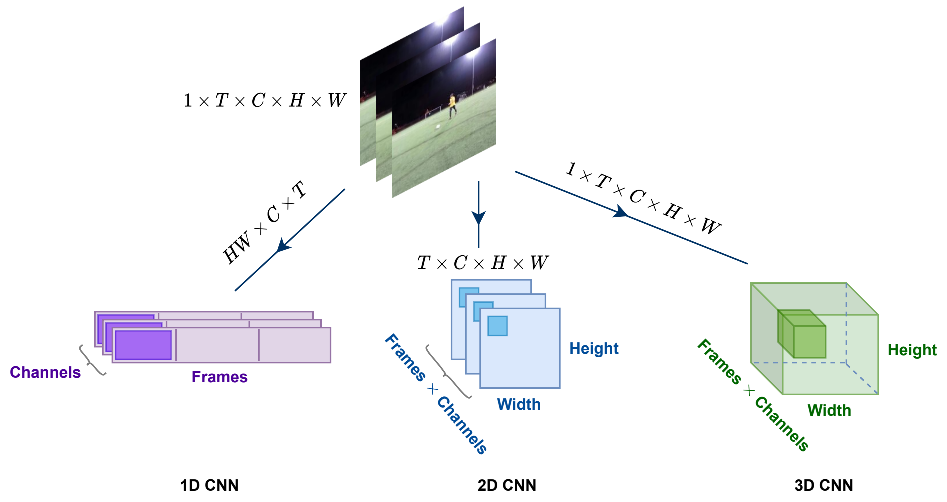


Fig. 2.10 An input of 4D image data tensor and three CNN layers with different kernel dimensions.

Recurrent Networks

Feed-forward and convolutional neural network models from earlier are given spatial data. Additionally, this dissertation expects information processing in temporal representation, such as the input of image sequences. This particular type of neural network layer that enables learning from sequential data was introduced by Rumelhart *et al.* [66]. Compared to the ConvNets and fully-connected layers that allow only working on one complete data point snapshot of a time, RNN is designated to handle temporal input structure $x_1 ... x_t$, e.g., text, speech, video, and time-series data by managing the weight of sequence of connected networks through the hidden state. To capture long-term temporal information from videos, one intuitive approach is to introduce the Long Short-Term Memory (LSTM) module as an encoder to encode the relationship between the sequence-illustrating deep features.

The LSTM module was introduced by Hochreiter *et al.* [32]. This type of network has a mechanism to update the past information with new incoming ones of a sequence from a variable length of the input. Also, this module can tackle the vanishing gradient issue. The guiding idea of this paradigm is to transmit numerous temporal processes from the past to the future while forgetting less significant occurrences. The fundamental idea behind LSTM is the memory cell, which distinguishes it from the idea behind RNN, which remembers everything forever and devalues new information as it comes in. Input, forget, and output gates make up an LSTM layer, as shown in Figure 2.11.

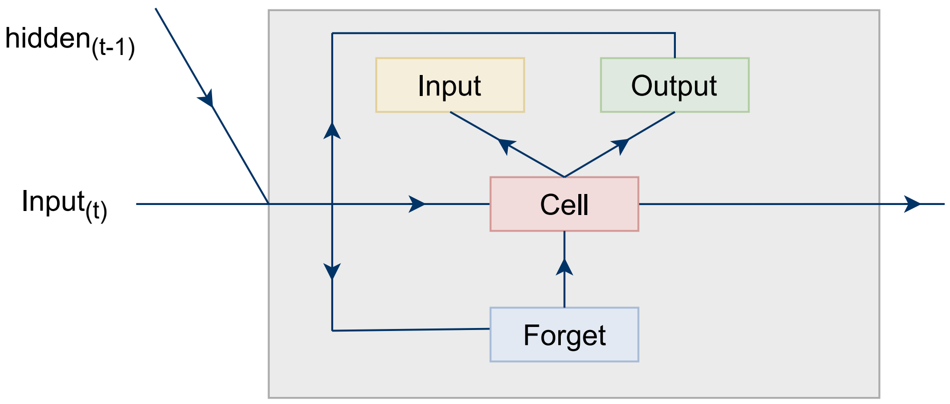


Fig. 2.11 An LSTM layer with units inside it.

The neural network model can delete unnecessary information using the forget gate. The LSTM module follows the rule in Equation 2.1. It has three kinds of input $[c_{t-1}, h_{t-1}, x_t]$ for learning the sequence feature representation. The explanation for each input is as follows: c_{t-1} represents a cell state from the previous step, h_{t-1} represents the output from the previous step, and x_t is the input at the current time step. These inputs then pass through upon the two sigmoids to obtain the input gate i_t and forget gate f_t.

$$\begin{aligned}
f_t &= \sigma(W_f[\,c_{t-1}, h_{t-1}, x_t]\ + b_f) \\
i_t &= \sigma(W_i[c_{t-1}, h_{t-1}, x_t] + b_i) \\
g_t &= \tanh(W_g[h_t, x_t] + b_g) \\
c_t &= f_t \times c_{t-1} + i_t \times g_t \\
o_t &= \sigma(W_o[c_{t-1}, h_{t-1}, x_t] + b_o) \\
h_t &= o_t \times \tanh(c_t)
\end{aligned} \tag{2.1}$$

Another type of LSTM is bidirectional LSTM (abbreviated as bi-LSTM). Bidirectional LSTM (BI-LSTM), a class of LSTM networks developed by Huang *et al.* [34], considers not only past learning but also considers future sequence when learning the sequence pattern. The forward direction input and reversed output of two stacked LSTMs form the BI-LSTM (see Figure 2.12). It has already been used to solve various real-world issues, including writing recognition and machine translation.

Regularization

Quoted from a book by Goodfellow *et al.* [22] in the context of deep learning, most regularization strategies are based on regularizing estimators. When an estimator is

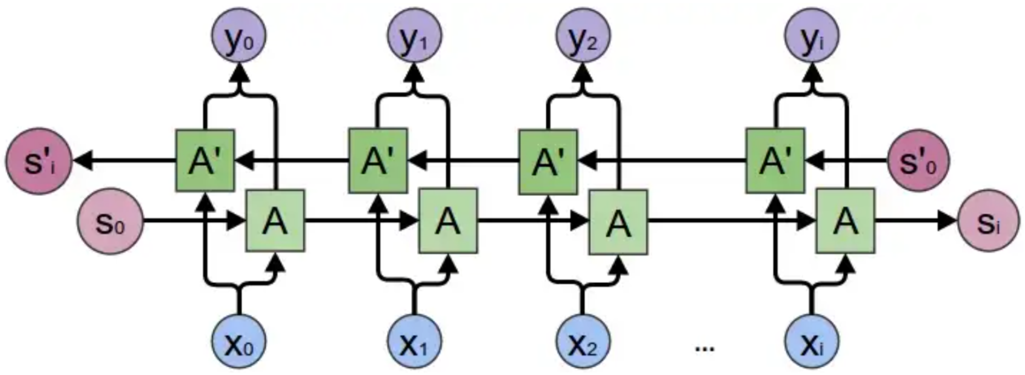


Fig. 2.12 A simplified bi-LSTM. Image is taken from a website[1].

regularized, it trades more bias for less variance. A regularizer is effective if it creates a profit, significantly reduces volatility, and does not excessively increase bias. Said regularization makes models simpler. Here, we will discuss a few deep-learning model regularization strategies.

- **Dropout.** Dropout falls into noise injection techniques and can be seen as noise injection into the hidden units of the network. In practice, during training, some number of layer outputs are randomly ignored (dropped out) with probability $p \in (0, 1)$. This probability is a hyperparameter and does not need to be the same for each layer. During inference time, all units are present, but they have been scaled down by p. This condition happens because the next layers will receive lower values after Dropout. An illustration of the dropout mechanism can be seen in Figure 2.13.

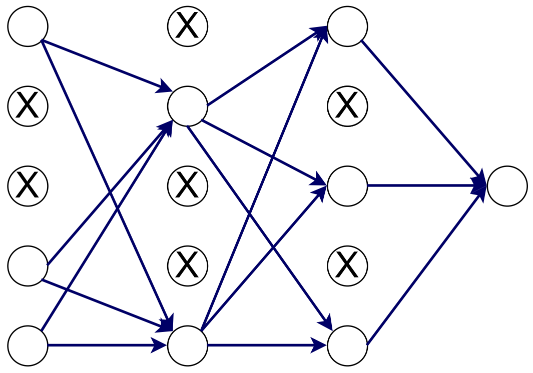


Fig. 2.13 An overview of a dropout mechanism. At every iteration, it randomly selects some nodes and removes them along with all their incoming and outgoing connections, as shown.

[1]https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0

The same layer will alter its connectivity by using Dropout and searching for alternative paths to convey the information in the next layer. Consequently, each update to a layer during training is conducted with a unique "view" of the specified layer. Conceptually, it approximates training many neural networks with different architectures in parallel. "Dropping" values means temporarily removing them from the network for the current forward pass, along with all its incoming and outgoing connections. Dropout has the effect of making the training process noisy. The choice of the probability p depends on the architecture.

- **Batch normalization.** Batch normalization (BN) can also be used as a form of regularization. Batch normalization fixes the means and variances of the input by bringing the feature into the same range. More specifically, values of feature maps are forced in a compact Gaussian-like space. When upgrading the weights in a network, all layers are updated, assuming the other layers do not change. When updating all weights simultaneously, unexpected results may occur due to this assumption. To reduce the effect an update to previous layers has on a future layer, batch normalization aims to normalize the output of the activations of each layer by introducing learnable variance and bias parameters γ and β. Let the matrix H represent the activations of one layer from a certain batch. Each row in H corresponds to the activations for an example in the batch. Then we define the mean and standard deviation of H as μ and σ. The output H is replaced by:

$$H' = \frac{H - \mu}{\delta} \tag{2.2}$$

The output data is now normalized; however, normalizing the output might reduce the expressive power of the network. To combat this, the previously mentioned learnable parameters γ and β replaces the output H' with:

$$H'\prime = \gamma H\prime + \beta \tag{2.3}$$

- **Data augmentation.** The simplest way to reduce overfitting is to increase the training data size. In machine learning and deep learning, it is not feasible to increase training data size as the labeled data is too costly. Data augmentation encompasses a suite of techniques that enhance the size and quality of training datasets such that better deep-learning models can be built using them. The image augmentation algorithms include geometric transformations, color space augmentations, kernel filters, mixing images, random erasing, feature space augmentation, and so on.

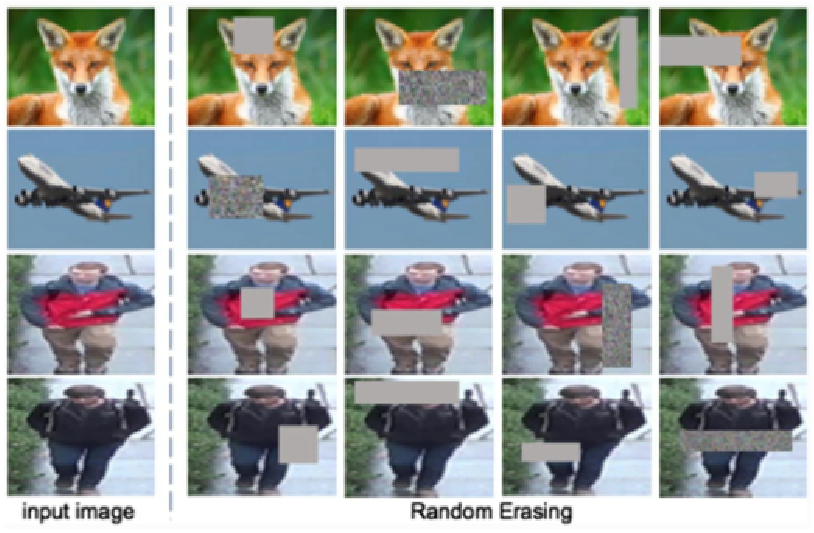


Fig. 2.14 Some examples of image augmentations. Notice that the transformed images are similar to the original images, except some parts of transformed images are masked out in order to build the deep learning model more general to the other unseen images.

For action recognition tasks, researchers usually use temporal augmentation, besides spatial augmentation. In short, temporal augmentation is regarded as selecting frames for training, e.g., using a sparse sampling technique.

Optimizers

When a network learns, it updates the weights to take a step toward reducing the loss. A standard way of doing this is with gradient descent. This method works by taking a step in the direction of the negative gradient of the loss function at the current weight value w and updating w with the new value. The new w_{n+1} is updated by:

$$w_{n+1} = w_n - \gamma \nabla L(w_n) \tag{2.4}$$

where γ is the step size (also called learning rate), w_n are the current weights, and $\nabla L(w_n)$ is the gradient of the loss function with regards to the weights.

Data processing

Before a network can be trained, the data must follow a specific structure, often limited to the required structure of the network. For example, the structure required for a

Resnet [31] is an image with three depth channels, often RGB-values, and a height and width in pixels. The height and width are recommended to be above 224×224 for the network to perform in the desired manner. A higher-resolution image might yield more information but requires more memory and longer computation time. Another processing step is normalization. This process is to normalize the values of the inputs to a set interval, often between 0 and 1. It will reduce errors due to significant value differences in the input.

Hyper parameters

When training a network, weights are parameters that are updated during the training. Hyperparameters are parameters that must be set prior to the training. These are typically the learning rate, batch size, or the algorithm used to minimize the loss function. It is hard to determine what values of the hyperparameters will yield the best model performance. A grid search is used to find the hyperparameters that work best for the current problem. It works by simply testing different combinations and seeing what works best [93].

Transfer Learning

Transfer learning is training a network to complete a task in a new but similar domain as the network is initially trained on. Figure 2.15 illustrates the transfer learning approach. For example, a network trained to recognize photos of cats and dogs can be taught to recognize butterflies. Transfer learning is frequently applied when a larger dataset can extract information pertinent to a new task, and there is little data in the new domain. There are typically two approaches to retraining a network to address a new challenge—shallow and deep retraining. The number of layers updated with the new data makes a difference between these. When training in deep retraining, the network's weights are all updated, whereas only the last layers are optimized in shallow retraining. Other layers' weights are frozen because low-and-middle-level characteristics, like edges or pixel intensities, are frequently extracted in a network's initial layers and are still valuable in the new domain. When the input of the new data differs greatly from the dataset, the network was previously trained on, deep retraining is performed.

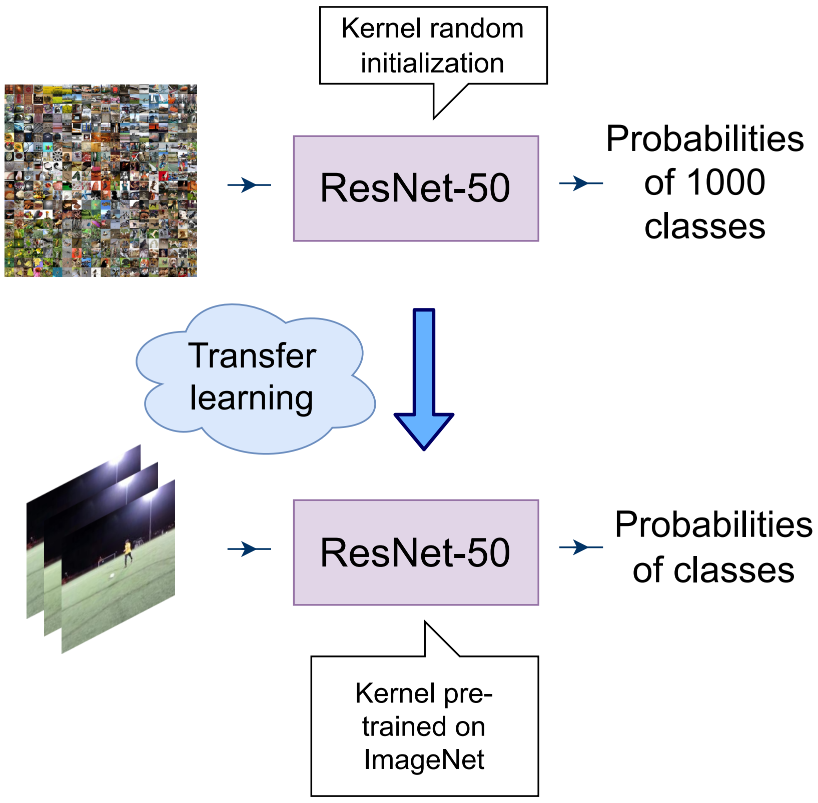

Kernel random initialization
ResNet-50
Probabilities of 1000 classes
Transfer learning
ResNet-50
Probabilities of classes
Kernel pre-trained on ImageNet

Fig. 2.15 Re-using ResNet-50 for a different task. First, a random kernel initialization of ResNet-50 is utilized for large-scale ImageNet dataset training. After the model has been optimized for ImageNet, a pre-trained model is used for an action recognition task with a smaller number of classes.

Learning new information and skills is one of a person's most crucial abilities. We can learn similar content more quickly without studying it from scratch by drawing on the prior knowledge stored in our brains and our studying experience. Humans transmit previously preprocessed information in their unique ways to learn related new information. A breakthrough in machine learning is the introduction of the transfer learning algorithm, which disproves the common assumption that a training dataset must come from the same source as a subsequent testing dataset, indicating that the distribution of the two datasets is identical. Transfer learning could help standard machine learning algorithms acquire new knowledge from future datasets from another distribution by reusing previously preprocessed data for those latter datasets collected from multiple distributions with distinct properties in similar tasks. This crucial feature lowers the cost of labeling new data, retraining new models, and computational resources for machine learning based on transfer learning.

Knowledge transfer across the linked source and destination domains is the aim of transfer learning. In other words, transfer learning extends model adaption and application in multiple related data and tasks with the same goals by applying information learned from the source material to another similar material in a new environment.

Feature Extraction

Feature extraction in neural networks contains the representations that are learned by the previous network to extract the interesting features from new samples, as illustrated in Figure 2.16. The features are then run through the new classifier, already trained from scratch.

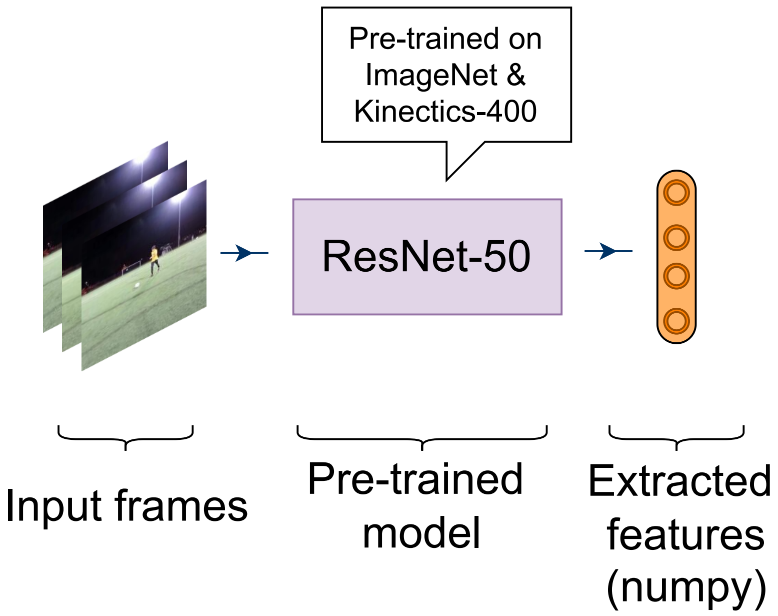

Pre-trained on ImageNet & Kinectics-400
ResNet-50
Input frames
Pre-trained model
Extracted features (numpy)

Fig. 2.16 An overview of the feature extraction process. The RGB images are fed into the deep learning model pre-trained on ImageNet and Kinetics and obtain meaningful features estimating the underlying action for the input frames. These features are usually saved as a Numpy array for further classification steps.

Feature extraction, in the case of the ConvNets, consists of taking the convolutional base of the previously trained network, running new data through it, and finally training the new classifier on top of the network's output. The first part of ConvNets, which contain a series of pooling and convolution layers and finally connects with a classifier, is called the convolution base of the model.

Feature extraction is generally used on convolution bases that are more generic than densely connected layers. Convolution bases are reusable also. The feature maps of the ConvNet are the presence maps of generic concepts over a picture which is beneficial regardless of the computer-vision problem or any other problem.

Major Architectures of Deep Neural Networks

ResNet. Residual networks (ResNet) is a classic CNN used as a backbone for many computer vision tasks. This model won the ImageNet challenge in 2015. Low, medium and high functionality and classifiers are extracted into layers in deep networks. ResNet mainly solves two fundamental problems in training deep neural networks: vanishing and exploding gradients. Fig. 2.17 illustrates the ResNet architecture with 50 layers deep, made up of 5 blocks, where each contains a set of convolution and max-pooling layers, followed by a skip connection. For other variants of ResNet (e.g., 18 layers or 101 layers follow the same architecture).

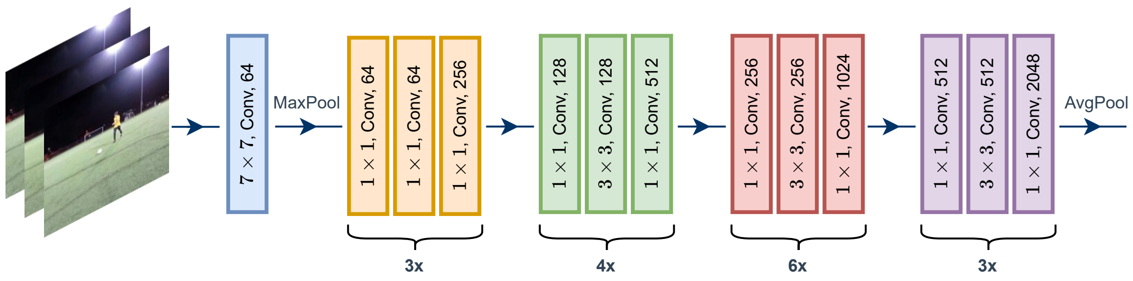


Fig. 2.17 A ResNet architecture with 50 layers.

The core idea of ResNet is to introduce the" identity shortcut connection," which skips one or more layers, as described in Figure 2.18. Skipping layers allows avoiding gradient to vanish during backpropagation. The phenomenon of vanishing gradient itself can occur when there are more layers in the network; at this point, the loss function's partial derivative reaches a value nearly equal to zero and vanishes. Using such connections, one can train extremely deep networks that can take advantage of the power of depth and hence allow capturing complex patterns in data without being concerned about the vanishing gradient problem.

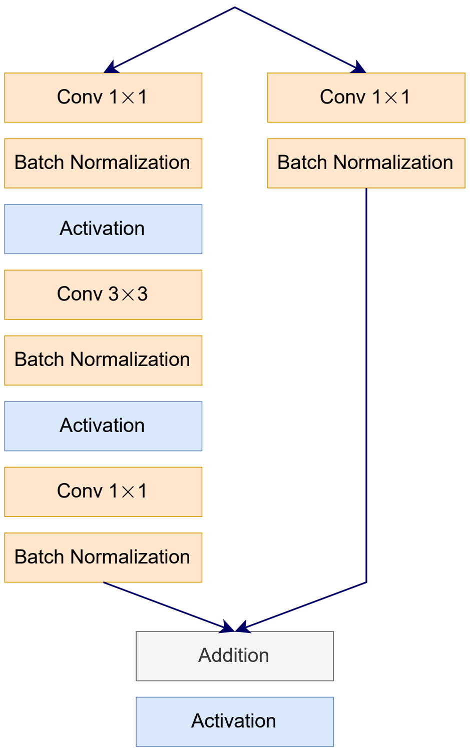


Fig. 2.18 A skip connection in each convolution block.

Inception. When performing convolutions, choosing between a 3×3 kernel size and a 5×5 kernel size for each layer is crucial. This choice needs to be made for every layer. The best action is to select the ideal layer mix for the entire network. This intuition motivated Inception, a cutting-edge architecture that introduces a multi-kernel size strategy. Configuring the network to have multi kernels will let the network choose the best configuration on its own.

In order to achieve this, we first perform each convolution in parallel on the same input with the same padding, meaning that the output's spatial dimensions match those of the input. Then, we combine the feature maps from each convolution into a single, large feature map. This process can be seen in Figure 2.19. The following conception module receives this substantial feature map as input. Although there is no practical limit to the number of filter sizes, the Inception architecture is limited to sizes of 1×1, 3×3, and 5×5. The smaller filters aid in capturing local details and features, whereas the larger filters focus on more dispersed, higher abstraction elements. Because pooling layers have been discovered to improve network performance, a 3×3 max pooling is also added to the Inception architecture.

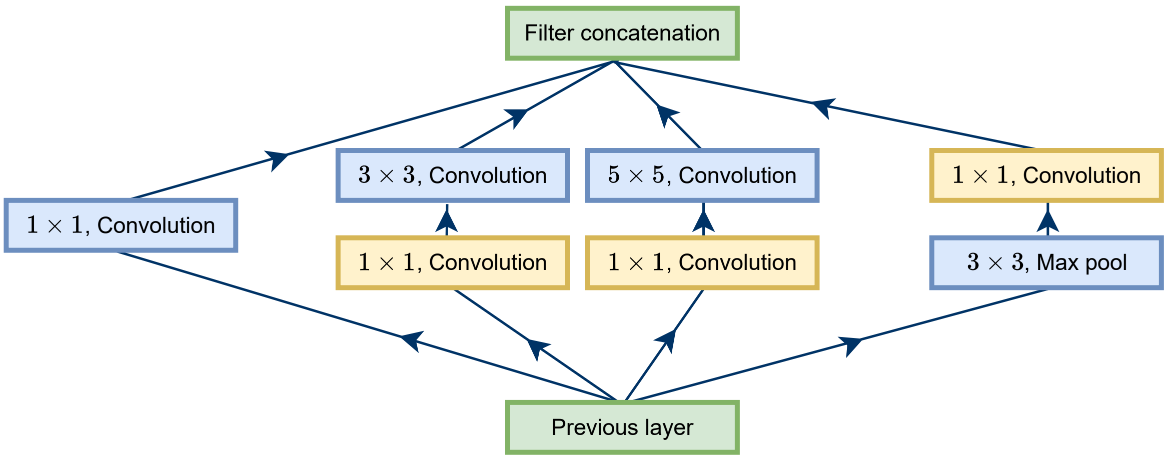


Fig. 2.19 An architecture of an Inception module with dimension reduction layers.

Inflated 3D. The I3D model was presented by researchers from DeepMind and the University of Oxford in a paper called "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset" [11]. The paper compares previous approaches to the problem of action detection in videos while presenting a new architecture, as shown in Figure 2.20.

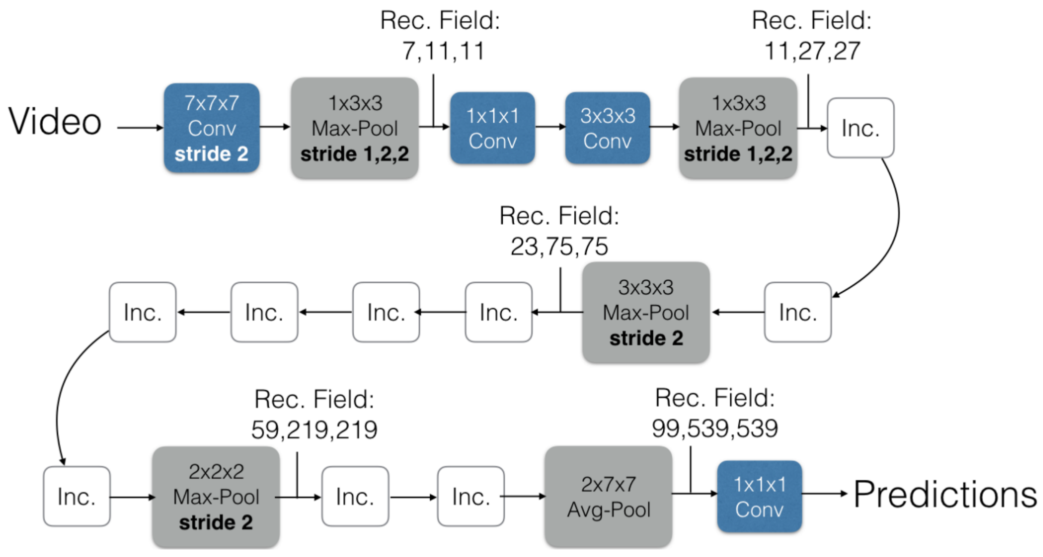


Fig. 2.20 The inflated Inception-V1 architecture. Image is taken from [11].

Their approach starts with a 2D architecture and inflates all the filters and pooling kernels. By inflating them, they add dimension to be considered, which in our case, is time. While filters in 2D models are square N×N, inflating them makes the filters become cubic N×N×N.

2.4 Application of Action Recognition

2.4.1 Criminal Detection in Video Surveillance

Burglary and automobile theft rates decreased between 1996 and 2006, per a report by Heiskanen *et al.* [29] in International Statistics on Crime and Criminal Justice. According to reports from Alexandrie [2] and Piza *et al.*, [61], this is primarily due to the dramatic growth in the use of monitoring technologies. However, this rapid growth also brings issues that must be resolved. In addition to the expense of establishing surveillance devices, monitoring the feed requires ongoing expenditures, which calls into question the usefulness of implementing surveillance for its stakeholders. Significant privacy concerns arise when monitoring surveillance. Governments and policymakers are hesitant to upgrade current surveillance systems due to the rise in labor costs and potential societal instability brought on by the erosion of individual privacy. According to the research [47], police officials claim that active surveillance monitoring and intervention prevent more crimes than they do. Better surveillance monitoring methods are required to protect privacy and provide quick crime detection for a prompt response. To meet the criteria mentioned above, the authors of this research suggest a real-time edge implementation for surveillance-based crime detection. Figure 2.21 illustrates the principle of criminal detection using surveillance cameras.

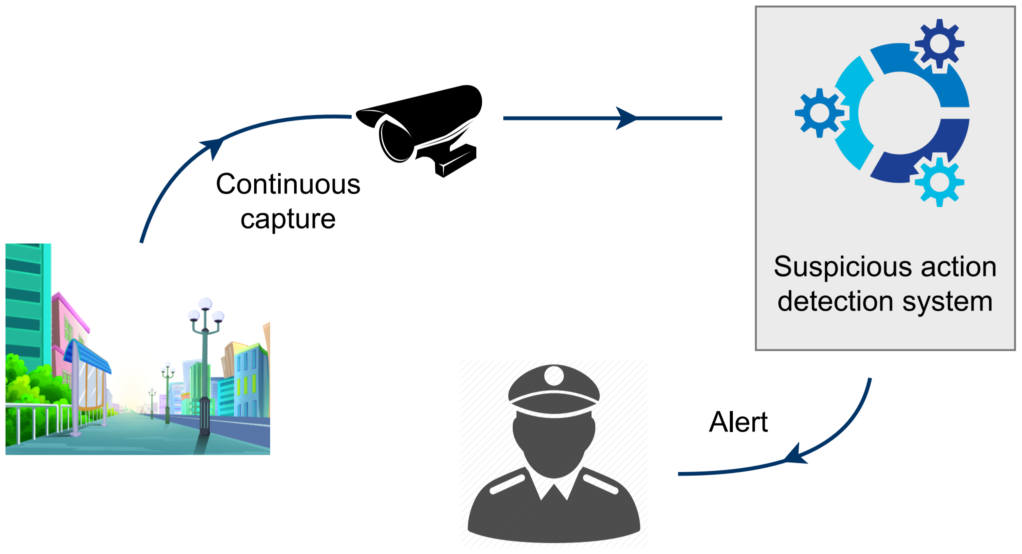


Fig. 2.21 An illustration of a suspicious activity detection system.

With potent machine learning and deep learning models on these devices, we have recently observed an enormous growth in edge-oriented research. The development of novel algorithms like ProtoNN [25], and Bonsai [46], which address the issue of real-time prediction on resource-constrained devices by significantly reducing the model size and inference time, has been made possible by machine learning algorithms like k-Nearest Neighbours (kNN) and tree-based algorithms. Meng *et al.* [56] presented a Two-Bit Network (TBN), which aids in the compression of large models like CNN, as an alternative for deploying computationally expensive models on resource-constrained devices. Multiple Instance Learning with Early Stopping is a novel approach that Dennis *et al.* [15] suggested to work on sequential data that may be utilized for edge implementation of deep models like RNN and LSTM.

2.4.2 Video Recommendation System

More than 150 million videos are available on YouTube, and 500 hours of new content are added every minute. Most movies are under ten minutes long, were shot by amateurs in poor lighting and image conditions, and contain little to no metadata. Before being used by a recommender system, this enormous volume of films must first be analyzed, categorized, tagged, and ranked by users or an artificial algorithm. The recommender system aids users with predetermined preferences in finding their desired material more quickly and navigating the large volume of visual content as efficiently as possible. Recommender systems maximize the valuable space devoted by websites to recommended material for their users by populating the landing page exclusively with content deemed relevant to a particular user. An illustration of the video recommender system is shown in Figure 2.22.

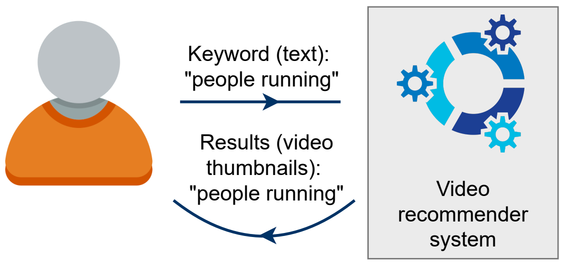


Fig. 2.22 An illustration of video-based recommendation system.

The ability to filter through enormous information spaces and choose the most likely to be exciting and appealing to a user distinguish Recommender Systems (RS) from other types of software [71]. Collaborative filtering, content-based, and hybrid methods are the three main categories under which recommendation techniques are typically categorized [37]. Popular content-based techniques propose products with content traits resembling those of products a consumer has previously liked [9, 7]. For instance, news suggestions look for similarities between words or keywords in articles. The availability of data regarding the content attributes of the objects is a requirement for content-based filtering. Most current systems correlate these qualities with the items as structured or unstructured meta-information. For instance, many RS in the film domain consider movie genre, director, actor, or (organized information), or narrative, tags, and textual reviews (unstructured information).

The main building blocks of recommendation systems are textual video data and video content similarity, which are then compared to user profiles and histories. In order to provide a list of the best matches, in the form of other videos the user is probably interested in, the video features are utilized to estimate the "distance" of a user profile from them. Generally speaking, video recommendation functions by mining user profiles and viewing history and then rating videos following those users' tastes and viewing patterns. Video suggestion is a ranking task because each video is ranked according to the user's search terms and phrases. Education, background, occupation, philosophy, location, interests, and other data about users (profiles) are collected. Each item must be grouped and assigned to a specific class using similarity criteria. The title, description, tags, and category are all included in the information for the video.

Nowadays, individuals may upload and distribute videos on the Internet thanks to the rapid development of technology. However, since most search engines employ the accompanying text data to handle video data, organizing and retrieving videos according to video content is a very difficult task [20]. Video retrieval may be unsuccessful if the text information, including tags, titles, descriptions, and keywords, is inaccurate, illegible, or irrelevant [65]. Analyzing human behavior in videos is an alternate approach because most recordings already have such a cue. As an illustration, by calculating the similarity of action representations, researchers developed a framework for retrieving videos [89]. In contrast to traditional human action recognition tasks, retrieval ranking is used instead of categorization in video retrieval tasks [45].

Chapter 3

Spatio-Temporal and Long-Term Sequence Features Learning

3.1 Introduction

Convolutional Neural Networks (CNNs) have proven remarkably effective at solving static image recognition challenges such as the MNIST, CIFAR, and ImageNet Large-Scale Visual Recognition Challenge. By employing a hierarchical structure of trainable filters and feature pooling processes, CNNs can automatically learn the complex features required for visual object recognition tasks, outperforming handwritten features. These encouraging results have prompted recent proposals for using CNNs to action categorization tasks in videos.

Video delivers more information than an image for a recognition challenge by incorporating a temporal component that enables the utilization of motion and other information. In addition, the operation is far more computationally intensive, even when analyzing small video snippets. Each video may include hundreds or thousands of frames, but not all are useful. An easy strategy is treating video frames as still images, applying 2D CNNs to each frame, and averaging the predictions at the video level. However, while these algorithms are more efficient than their 3D equivalents, they cannot infer temporal order or more complex temporal relationships. Consequently, it is natural for 3D CNNs to be able to learn spatial and temporal aspects for action detection tasks.

Meanwhile, Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) have effectively handled sequential multimedia data and produced cutting-edge outcomes in speech recognition, digital signal processing, video processing, and text data analysis. Later, LSTM is applied more frequently to learn sequential information among frame features.

Action recognition tasks may benefit from combining 2D CNN and 3D CNN with LSTM. In this chapter, we proposed an I3D-DenseLSTM hybrid deep learning architecture for describing action recognition features. To construct a hybrid model, multiple deep neural networks were integrated to predict a single outcome. We employ information fusion approaches to mix data from many sources to recognize a new view for improved action detection classification.

The rest of the chapter is structured as follows: **Section 3.2** describes work related to the proposed method. **Section 3.3** explains in detail our composite model. **Section 3.4** includes experiments and evaluation as well as the comparisons with the state-of-arts to show the effectiveness of our proposed method. To conclude the chapter, we present a summary of our proposed approach in **Section 3.5**.

3.2 Related Work

3.2.1 Spatial and Temporal Learning

CNN, a family of deep models for constructing features, has been chiefly applied to 2D images. Treating video frames as still images and applying CNNs to distinguish individual frame-level activities is a straightforward technique in this area. However, this method disregards the motion information stored in numerous consecutive frames. To successfully incorporate motion information into video analysis, we propose implementing a 3D kernel in the base network so that spatial and temporal discriminative characteristics are collected.

Based on our knowledge, the first attempt to include temporal dimension along with spatial dimension was introduced by [39] and later adopted by [76]. In their work, they apply an extension of 2D convolution to include a temporal dimension. In 2D CNNs, convolutions are applied to the 2D feature maps only to compute features from the spatial dimensions. When applied to video analysis problems, capturing the motion information encoded in multiple contiguous frames is desirable. To capture the temporal dimension as well, they propose to perform a 3D kernel in the convolution stages of CNNs to compute features from both spatial and temporal dimensions. The 3D convolution is achieved by convolving a 3D kernel to the cube formed by stacking multiple contiguous frames together. By this construction, the feature maps in the convolution layer are connected to multiple contiguous frames in the previous layer, thereby capturing motion information. Formally, the value at position (x,y,z) on the jth feature map in the ith layer is given by:

$$v_{ij}^{xyz} = b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)} \tag{3.1}$$

where R_i is the size of the 3D kernel along the temporal dimension, w_{pqr}^{ijm} is the (p,q,r)th value of the kernel connected to the mth feature map in the previous layer.

Another work by [28] conducted various experiments using the Kinetics, UCF-101, and HMDB-51 datasets to explore good practices for training 3D CNNs. They focus on configurations of data augmentation and finetuning for 3D CNNs. Several data augmentation configurations for videos, such as spatiotemporal random cropping and spatial corner cropping, are used by different researchers. Their experiments compare these configurations and show the best configuration for 3D CNNs. They then compare finetuning configurations. Most works for 3D CNNs train all layers in finetuning, whereas some works report that freezing early layers and training only top layers achieved the best performance [28].

A work by [27] investigated using a residual network architecture for action recognition. Deep 3D CNNs for action recognition has yet to be explored enough because of the training difficulty caused by many of their parameters. Prior work in image recognition shows that very deep architectures of CNN's improve recognition accuracy [31]. Exploring various deeper models for the 3D CNNs and achieving lower loss at convergence is essential to improve action recognition performance. Applying the architecture of ResNets to 3D CNNs is expected to contribute to further improvements in action recognition performance. In this dissertation, we experimentally evaluate 3D ResNets or their variants to get good models for action recognition. In other words, the goal is to generate a standard pre-trained model in spatiotemporal recognition.

3.2.2 Long-Term Sequence Learning

Action recognition using video analysis is computationally expensive. Short video processing may take a long time due to its high frame rate. As each frame plays an essential role in a video story, keeping information on sequential frames for a long time makes the system more efficient. The proposed method uses a recurrent neural network, LSTM, to analyze the frame-to-frame change of action videos. RNNs are building blocks of a connected neuron with input units, internal (or hidden) units, and output units, having an activation at time t, which can selectively process data in sequence. As it processes one element at a time, it can model outputs consisting of a sequence of elements that are not independent. The RNN architecture strengthens processing and finding hidden patterns in time-space data such as audio, video, and text. RNN processes data in a sequential way such that at each time t, it gets input from the previous hidden state S_{t-1} and new data x_t. The data is also multiplied with weights, biases are added, and is fed to activation functions. Due to a large number of calculations, the effect of the initial inputs becomes negligible for the upcoming data sequence after a few layers, resulting in a vanishing gradient problem. The solution to this problem is LSTM. The main idea of LSTM architecture is its memory cell, input gate, output gate, and forget gate, which can maintain its state over time T_N, and non-linear gating units, which regulate the information flow into/out of the cell. Researchers have presented different variations of LSTM, such as multi-layer LSTM and bidirectional LSTM, for processing sequential data as in Figure 3.1.

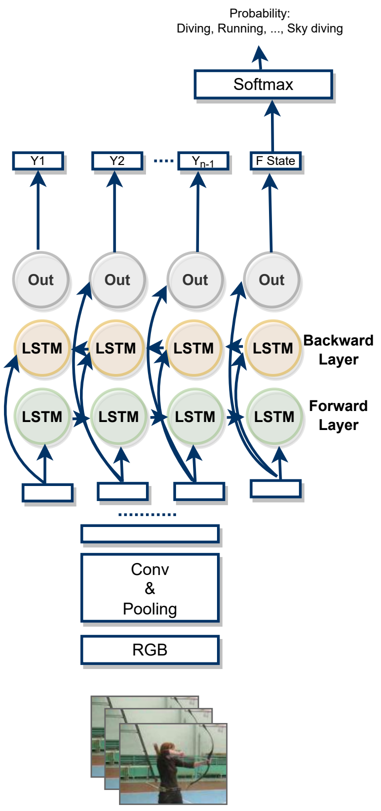


Fig. 3.1 A network with bi-LSTM on top of the base network to analyze hidden sequential patterns in both temporal and spatial sequential data.

The proposed method analyzes the complex pattern in the visual data of each frame, which cannot be efficiently identified using simple LSTM and multi-layer LSTM. In the proposed method, features of video frames are analyzed for action recognition. Deep features from every sixth frame of a video are extracted using pre-trained AlexNet. Next, an architecture of DB-LSTM is developed with two layers at each forward and backward pass for learning sequence information in the features of video frames. The proposed method can recognize actions in long videos because the video is processed in N time steps.

3.2.3 Multi-Stream Architecture

A video comprises a series of still-image (spatial) frames where all frames are densely stacked along a temporal dimension. Thus, it is common to learn both spatial and temporal representations from a video to predict an action involved in a video. Other researchers also studied the benefit of optical flow data to enhance the model's performance. The optical flow refers to the visible motion of an object in a series of images and the apparent 'flow' of pixels in an image; thus, it will describe a motion cue of an object. In this section, we will briefly discuss several works related to the proposed method, including work from Simonyan *et al.* [70], Ullah *et al.* [80], and Feichtenhofer *et al.* [18].

Simonyan *et al.* [70] (Figure 3.2a) proposed a two-stream CNN architecture from RGB video sequences and computed multiple optical flows. They introduced a structure dependent on spatial and temporal streams. The spatial stream performed action recognition from video sequences. Simultaneously, the temporal stream of CNN trained on dense optical flow volumes and saved the horizontal and vertical displacement vectors from successive action frames, i.e., recognizing action from motion using dense optical flow. Finally, a temporal CNN and a spatial CNN were fused using the softmax scores.

A work by Feichtenhofer *et al.* [18] studied several strategies for combining spatial and temporal cues at various levels of granularity in feature abstraction. They proposed a novel architecture for the spatiotemporal fusion of two stream networks that achieves state-of-the-art performance (see Figure 3.2).

3.3 Proposed Multi-Stream with Composite of I3D and LSTM

We propose a novel framework combining the temporal model and convolutional perceptual representation. In detail, we propose a three-stream network, each stream

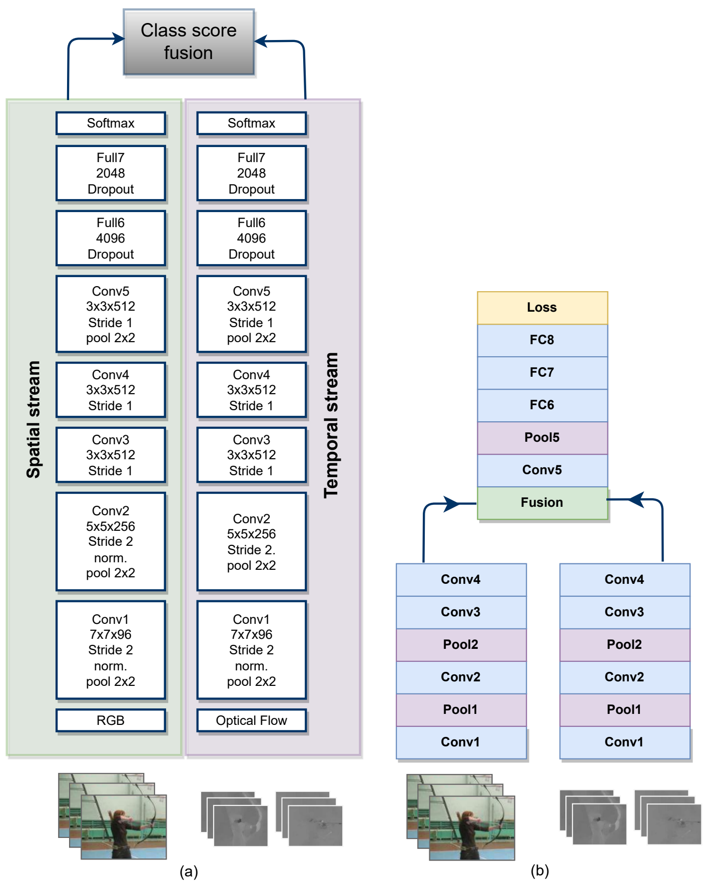


Fig. 3.2 Two related work of video classification from Simonyan *et al.* [70] and Feichtenhofer *et al.* [18]. **(a)** Two-stream architecture to accommodate the spatial and temporal aspects of a video. The spatial stream allows the network to identify an object related to the underlying action; the temporal stream is designated to learn motion representation in encoded optical flow data. **(b)** Two-stream architecture for video classification from [18].

has different network architecture. The proposed model is an ensemble model for data diversity. In this section, we will explain each stream individually.

3.3.1 Convolution Block with I3D

In this stream, the input raw is optical flow. The motion information represented in the optical flow is often complementary to the raw video data for video tasks. We utilize a 3D version of Inception to extract motion features of optical flow data. We take the benefit of transfer learning, a common practice in deep learning, by re-using the pre-trained model's parameters on the Kinetics dataset to the I3D base model. The I3D network is the 3D version of the Inception network (see Figure 2.20).

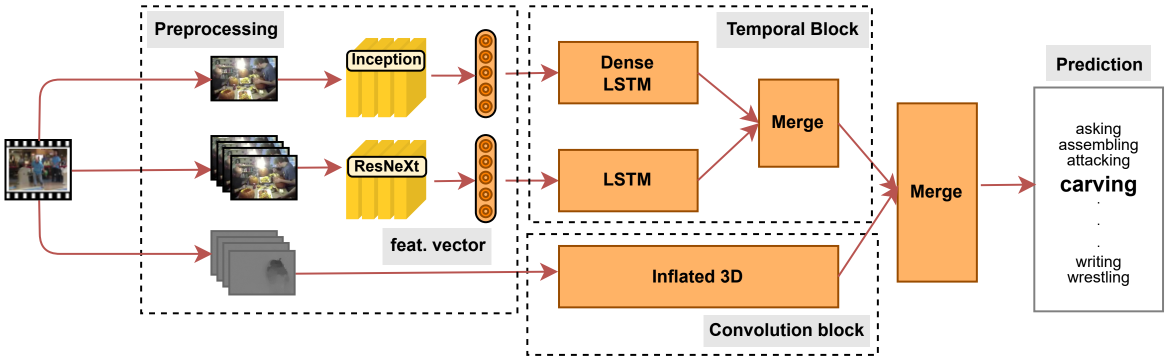


Fig. 3.3 Overview of multi-stream video classification architecture, consisting of temporal and convolution blocks.

3.3.2 Temporal Block with LSTM

In our proposed architecture, we utilize long short-term memory (LSTM) for modeling long-term motion information of the "Feat. 3D" branch. The LSTM itself has a memory cell component that preserves essential information. It has a mechanism to forget unimportant information rather than remember everything forever.

We utilize bidirectional LSTM (bi-LSTM) for the 2D CNN branch. In bi-LSTM, the output at time t is dependent on the sequence's previous frames and the upcoming frames. Bidirectional RNNs are pretty simple, with two RNNs stacked on top. One RNN goes in the forward direction, and another in the backward direction. The combined output is then computed based on the hidden state of both RNNs. We use multiple LSTM layers in our work, so our scheme has four LSTM layers for both forward pass and backward pass. Fig. 4 shows the overall concept of bi-LSTM used in the proposed method. This module consists of multiple bi-LSTM layers. For the first bi-LSTM layer, the input is a feature vector sequence from the 2D Inception network. The output is $h^1 = h_1^1, h_2^1, \cdots, h_s^1$, in which h_t^1 is formulated by the following equation:

$$\begin{aligned} \overrightarrow{h_t^1} &= lstm(\overrightarrow{h_{t-1}}, e(w_t)), \\ \overleftarrow{h_t^1} &= lstm(\overleftarrow{h_{t-1}}, e(w_t)). \end{aligned} \tag{3.2}$$

The second bi-LSTM layer's input is from the former bi-LSTM layer, and the output is $h^2 = h_1^2, h_2^2, \cdots, h_s^2$. The subsequent Bi-LSTM layers are processed accordingly. We can define the above process as follows:

$$
\begin{aligned}
h_t^l &= [\overrightarrow{h_t^l}; \overleftarrow{h_t^l}], \qquad h_t^0 = \text{2D Inception feature vector} \\
\overrightarrow{h_t^l} &= lstm(\overrightarrow{h_{t-1}^l}, M_t^{l-1}), \\
\overleftarrow{h_t^l} &= lstm(\overleftarrow{h_{t+1}^l}, M_t^{l-1}), \\
M_t^{l-1} &= [h_t^0; h_t^1; \cdots ; h_t^{l-1}]
\end{aligned}
\tag{3.3}
$$

3.4 Experiments and Evaluation

This subsection will describe how the experiment and evaluation are conducted. The experiment and evaluation are conducted to verify the effectiveness and impact of the proposed framework.

3.4.1 Dataset

This experiment uses the Moment In Time (MIT) Mini Track version. This dataset contains 110,000 videos consisting of 100,000 training videos and 10,000 validation videos. Some videos are equipped with audio. This version of the dataset has 200 verbs ranging from arresting to yawning. Each video has 3 seconds in length and operates at 30 frames per second. This dataset offers a balanced video distribution, meaning that every class or verb has the same amount of video.

Moreover, the agents are not just humans but can be animals, inanimate objects, and natural phenomena. The dataset we used for preliminary research is a collection of one million short videos, each with a label corresponding to an event unfolding in three seconds. Additionally, three seconds is a temporal envelope that holds meaningful actions between people, objects, and phenomena (e.g., wind blowing, objects falling on the floor, shaking hands, playing with a pet, and so on). Another challenge is that sequences of three-second actions can represent compound activities at longer time scales. For example, picking up an object and running could be interpreted as the compound actions "stealing", "saving", or "playing sports", depending on the context of the activity (e.g., agent and scene).

3.4.2 Evaluation Measure

To evaluate the effectiveness of our proposed multi-stream method, i.e., the model's accuracy can correctly predict, we used two standard evaluation measures for action recognition, including Top-1 and Top-5 accuracies. Top-1 accuracy is the conventional accuracy that the model answer (the one with the highest probability) must be exactly the expected answer. Meanwhile, Top-5 accuracy is the fraction of test samples containing at least one ground truth label in the top 5 predictions. Top-k accuracy is calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.4}$$

3.4.3 Implementation

Firstly, we extract all frames from all videos. Then, we extract features from RGB frames using 2D and 3D-based models. For the 2D-based extractor, the model uses Inception pre-trained on large-scale dataset ImageNet. Inception uses convolution works on 2D space and employs multiple sizes of filters on the same level to catch salient parts in the image better. Before feeding frames into Inception, we perform spatial transformation first. We sample only 21 frames uniformly to preserve the diverse distribution. Then we resize the frame from 256× 256 to 224× 224. We also scale frames to have a value between -1 and 1. Then these scaled frames are then fed into feature extractors as described in Figure 3.4. We do not perform any other data augmentation or pre-processing, such as subtracting the mean for each channel in RGB or flipping images. The extracted feature has a dimension of 1024 after applying PCA with whitening.

In the next step, for the 3D-based extractor, we employ 3D ResNext [33] with 101 layers pre-trained on Kinetics 400 dataset. As opposed to Inception, ResNeXt uses 3D convolutional, which takes multiple frames to process. ResNeXt is an advancement over the popular ResNet model. It added a cardinality dimension on top of the ResNet module. We sample the first 60 frames per 3-second video to extract spatiotemporal features in the temporal block (see figure 2). Like the pre-processing technique used in the 2D-based extractor, we perform similar steps to the frames first before feeding them to 3D ResNeXt. The sampling strategy can be extended to overlapping sampling to capture a temporal aspect of a video better. Moreover, sampling all frames equidistantly from the beginning until the last frame is beneficial since the event or activity can occur at any specific time. The extraction process yields 2048-d features of shape.

Meanwhile, we utilize densely connected Bidirectional LSTM (Bi-LSTM) for the "Feat. 2D" branch. In Bi-LSTM, the output at time t is dependent on the sequence's previous frames and the upcoming frames. Bidirectional RNNs are pretty simple, with two RNNs stacked on top. One RNN goes in the forward direction, and another in the backward direction. The combined output is then computed based on the hidden

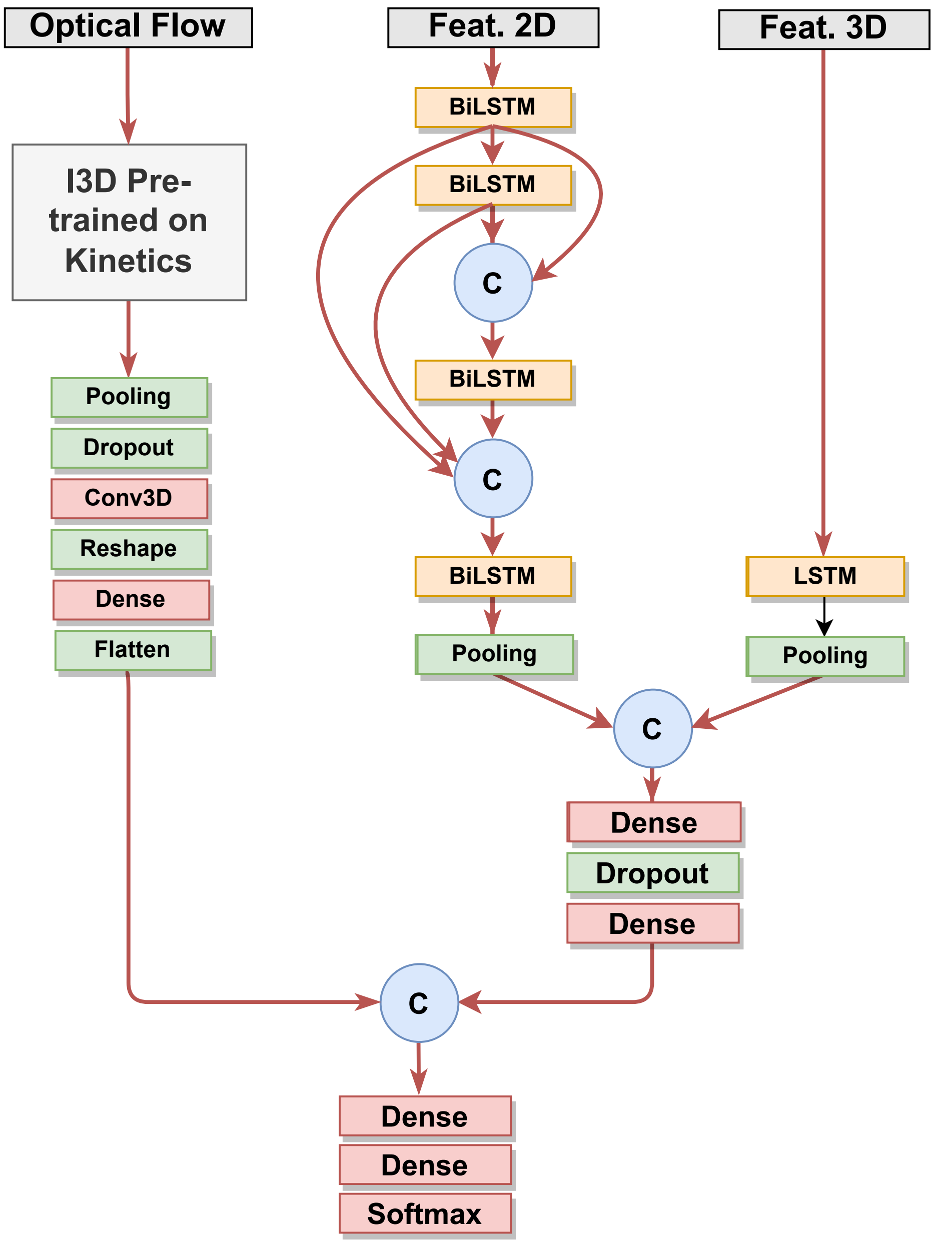


Fig. 3.4 Detail of the proposed model.

state of both RNNs. We use four Bi-LSTM layers in our work, so our scheme has four LSTM layers for both forward and backward passes.

Lastly, all streams are trained separately using Adam optimizer with a combination of step decay and constant learning rate. We implement step decay for the learning rate for the temporal block because we want to drop the learning rate after a specific interval as the training progresses. By dropping the learning rate to a predefined value after some epochs, we expect the learning process to achieve global optimum effectively.

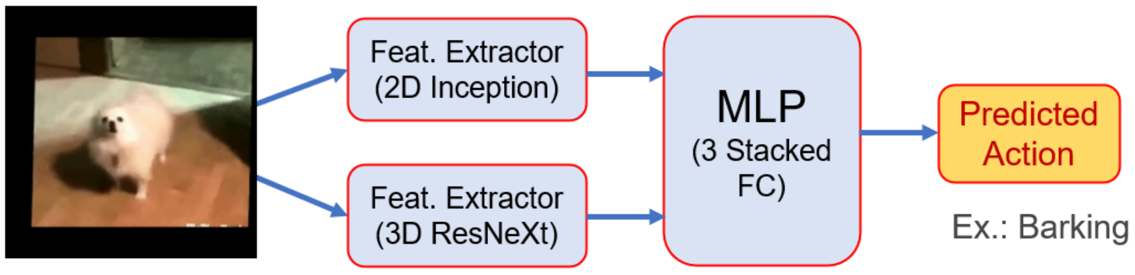


Fig. 3.5 The baseline network.

Regarding the baseline network, we design a naive two-stream network that consists of 2D Inception and 3D ResNeXt as feature extractors of a clip of video and are followed by three MLP with softmax activation function to calculate a multinomial probability distribution (see Figure 3.5).

3.4.4 Result & Analysis

In this subsection, we will explain our experimental result. We first compare each stream of our proposed model to test the effectiveness of integrating temporal and convolutional blocks in one network. We evaluate each stream using a validation set of the MIT mini-track dataset. Table 3.2 shows that a mixture of different models outperforms any other models. By combining all models into a single network, we can significantly increase the accuracy to 26.53% in top-1 accuracy and 52.41% in top-5 accuracy.

We then perform post-analysis on the confusion matrix using the combined network to distinguish different types of confusion. In addition, we examine which classes show good and poor accuracy. Here, we itemize different types of confusion:

- **Similarity in appearance.** We notice that some videos in one class share similarities with others in another. For example, *bicycling* and *pedaling*.
- **Semantic similarity.** We observe that some videos of *jumping* have similar meaning as in *bouncing*. It is hard to even for humans to differentiate this similarity.
- **Order in reverse.** Define an action or event with the same "movement" but in a different temporal order. *opening* and *closing* are the concrete examples of this confusion.

Table 3.1 F1 score of classes showing Top-5 best and Top-12 worst.

Classname	F1 Score	Remark
juggling	0.66	Top-5 best
shredding	0.64	
tattoing	0.63	
dunking	0.61	
erupting	0.60	
catching	0.03	Top-12 worst
covering	0.03	
falling	0.03	
feeding	0.03	
hitting	0.03	
pulling	0.03	
removing	0.03	
riding	0.03	
rising	0.03	
shouting	0.03	
destroying	0.03	
spraying	0.03	
Macro Average	0.26	All classes

From Table 3.1, we can see that some classes are categorized as good in performance, indicated by their F1 score are 60% or above. In contrast, several classes are poor since its score fall below 4%.

3.4.5 Comparison with Related Work

We will compare our work with another model's performance in several papers. Table 3.3 shows the comparison result. Guan *et al.* [24] use Pseudo-3D Residual Network [62] pre-trained on the Kinetics-400 dataset. This strategy is also adopted by [10].

3.4.6 Discussion

This chapter presents our preliminary research on action recognition tasks utilizing RGB and optical flow data and adopting multi-stream architecture. The two-stream architecture was proposed to take into account spatio-temporal information and optical flow, which boosted performance significantly compared to the one-stream architecture. The advantage of a two-stream network structure is that two convolutional neural

Table 3.2 Performance of each proposed model on a validation set of Moments in Time Mini dataset

Models	Modality	Top-1 Acc (%)	Top-5 Acc (%)	△ Top-1 (%)	△ Top-5 (%)
Baseline	RGB	23.78	46.76	–	-
2D Inception + DenseLSTM (A)	RGB	17.06	38.65	–	–
3D RestNeXt 101 + LSTM (B)	RGB	22.53	46.58	+5.47	+7.93
Inflated 3D (C)	Optical Flow	21.64	45.69	+4.58	+7.04
A + B	RGB	24.70	49.86	+7.64	+11.21
A + B + C (I3D-DenseLSTM)	RGB + Optical Flow	**26.53**	**52.41**	+9.47	+13.76

Table 3.3 Comparison of model performance with other methods on a validation set.

Models	Top-1 Acc (%)	Top-5 Acc (%)
P3D-Kinetics [24]	26.34	–
P3D [10]	14.70	33.40
TRN [10]	26.10	48.50
Our proposed method	**26.53**	**52.41**

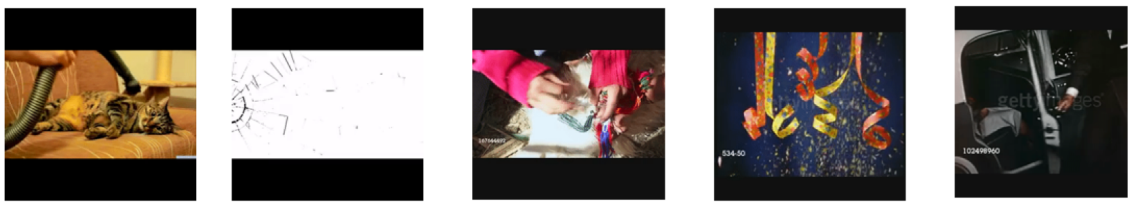
networks are used to obtain different action video features. It can increase the diversity of action description characteristics and obtain more discriminative action characteristics. Nevertheless, the learning of spatial and temporal features is isolated, making one stream is *unaware of the representation* learned by the other stream.

We equipped the framework with 3D CNN as a feature extractor. The 3D-CNN method increases the dimension of data input. The 3D convolutional network is practical and suitable for video spatiotemporal feature extraction. In the 3D CNN network, 3D convolution has more time dimension than 2D convolution; thus, it is suitable to adopt it for video action recognition. It is a primary network to extract visual appearance and short-time action features in complex human action recognition. However, 3D CNNs suffer from problems including *overfitting* and *slow convergence*. Also, 3D CNN introduces *heavy computations* inherent in 3D CNN-based frameworks that contribute to slow inferences.

Regarding the optical flow modality, even though the use of it can significantly increase the accuracy, the computation of optical flow is *time-consuming and storage demanding*. Thus, it is unsuitable for real-time application and should be avoided when considering fast inference.

In addition, we also implement LSTM on top of the RGB stream. The purpose of implementing LSTM over RNN is that the former has introduced the concept of cell states, which provide "highways" for the gradient to flow backward through time freely, thereby making it more resistant to the vanishing gradient problem. One major drawback of LSTM is math complexity for backpropagation; hence *extra time* for kernel weight optimization.

We also include some prediction results in this section, illustrated in Figure 3.6. In the first column, we successfully predict the sequence of frames as "*vacuuming*" with a probability score of 48.9%. Even though we cannot precisely predict correct answers in the second and third columns, our predictions are still in the top-5 range. For the rest



Predictions:	**Predictions:**	**Predictions:**	**Predictions:**	**Predictions:**
vacuuming: 0.4899	snowing: 0.2490	stitching: 0.2176	twisting: 0.1675	pressing: 0.2234
sanding: 0.2959	cracking: 0.1230	sewing: 0.1840	spinning: 0.0906	steering: 0.1557
cleaning: 0.0730	covering: 0.1111	knitting: 0.1075	bubbling: 0.0591	closing: 0.0968
washing: 0.0280	ascending: 0.0422	crafting: 0.0615	ascending: 0.0554	typing: 0.0967
removing: 0.0211	drawing: 0.0392	tying: 0.0594	combusting: 0.0536	entering: 0.0443
Actual: vacuuming	**Actual:** cracking	**Actual:** tying	**Actual:** dropping	**Actual:** opening
Result: Top 1 correct!	**Result:** Top 5 correct!	**Result:** Top 5 correct!	**Result:** Fail	**Result:** Fail

Fig. 3.6 Some prediction result on MIT validation set.

of the column, our proposed model cannot predict the actual labels of "*dropping*" and "*opening*" for column 4 and column 5, respectively. Instead, the model predicts the last two columns are of "*twisting*" and "*pressing*". If we closely observe the fourth image, it also twists when the object drops. For the fifth image, the model cannot reasonably make temporal reasoning. If we reverse the order of frames, our model's top 5 highest probability might contain the expected answer.

3.5 Summary

In this preliminary attempt at video action recognition, we successfully implemented a multi-stream neural network named I3D-DenseLSTM. The proposed method mainly consists of three streams: two are grouped as temporal blocks, and another is a convolution block. Each stream in the temporal block has different top layers: "Feat. 2D" stream has densely connected bi-LSTM, and "Feat.3D" has a vanilla LSTM layer. These two streams' outputs are then merged, followed by a linear layer. The convolution block has only one stream ("Optical flow" stream) with 3D convolution and a linear layer on top of the Inception 3D network. Output from the convolution block is then merged with output from the temporal block and fed to two linear layers with softmax activation function to get the class probability.

This approach is not end-to-end learning, meaning the proposed pipeline is designated to learn only some of the features between the original input (x) and the final output (y). Instead, the pipeline is divided into two steps; step one is to extract the features from RGB and optical flow data; step two is the learning process, as described in Figure 3.4.

Some limitations of this preliminary approach exist, as discussed in the previous section. In the subsequent chapter, we explore solutions to the problems mentioned earlier.

Chapter 4

Motion and Multi-view Excitation With Temporal Aggregation

4.1 Introduction

In **Chapter 3**, we have shown that we successfully build a framework to solve the multi-class classification problem of video action recognition. We presented a multi-stream approach consisting of three streams and accepted RGB and optical flow images as input. In addition to discussing the benefits and drawbacks of multi-stream architecture, we have also examined its advantages. This chapter will discuss these restrictions and present an innovative architecture for solving a multi-class classification problem.

A video action can be recognized based on scenes with less temporal information, while other actions need more temporal aspects to recognize. Such examples of actions with less temporal cues are *'rafting'* and *'haircut'*. We can judge the actions mentioned above only by seeing the scene. Contrarily, more temporal information is needed to judge an action involved in a video, e.g., *'zooming in with two fingers'* and *'picking something up'*. With this condition in mind, one must consider having spatiotemporal and motion information flow in the network.

Current existing convolutional neural networks (CNNs) for action recognition can be categorized by the type of convolution kernel, i.e., three-dimensional (3D) and two-dimensional (2D) CNN. Several researchers utilized 3D CNN to learn both spatial and temporal information simultaneously [76, 72, 40, 8, 27]. This method performs exceptionally well for video action recognition tasks; however, adopting a CNN-based 3D kernel adds more parameters than a 2D kernel, hence increasing the computation cost. This costly processing will restrict the application's real-time implementation. Besides, 3D CNNs are typically laborious to optimize and prone to overfitting[27].

To overcome the real-time and optimization problem, recently, more researchers utilized 2D CNNs and equipped the network with temporal convolution for characterizing the temporal information[43, 84, 100, 53]. TSN [86] employs a sparse temporal sampling strategy from the whole video to predict action, and this approach influenced several 2D CNN methods afterward. The approach by TSM[53] utilized a shift-part strategy together with a sampling strategy to recognize an action effectively. The shifting strategy is to shift part of the channels along the temporal axis to endow the network with movement learning. However, both approaches lack motion representations and spatiotemporal cues from different views. Meanwhile, many researchers consider this modality since optical flow[36] represents a motion as an input-level frame. Two main drawbacks of optical flow are that it needs vast space storage and a tremendous amount of time for computation which is not suitable for real-time application.

In different approaches from previous methods, a work by Tran *et al.* [77] factorized the 3D cube of a kernel into a (1+2)D kernel configuration to lessen the heavy computation. The author decomposed spatial and temporal modeling into two separate steps. Another interesting factorization strategy was introduced by the Deep HRI team[49] in the action recognition competition. They proposed their novel architecture, coined as multi-view CNN (MV-CNN), to act as a 3D convolution and showed a profound increase in accuracy. Nevertheless, these factorization strategies still need to pay more attention to the importance of motion features beneficial for action recognition tasks.

In this chapter, we have proposed a novel building block, **M**otion and multi-view **E**xcitation and **T**emporal **A**ggregation (META). Specifically, META comprises three submodules (1) Motion Excitation (ME), (2) Multi-view Excitation (MvE), and (3) Densely Connected Temporal Aggregation (DCTA). These three submodules are integrated into the 2D ResNet-50 base model, allowing the network to learn spatiotemporal and motion features jointly. The ME submodule handles optical flow difficulties by computing feature-level content displacement on the fly during training or inference, thus eliminating the requirement for space storage. It also introduces an insignificant amount of FLOPs and time to calculate compared to optical flow. The MvE submodule, on the other hand, produces an enhanced spatiotemporal representation of output features. This submodule adds a multi-view perspective to the original feature maps. We design MvE submodule to be complementary to ME in that output from MvE is directly added to the output from ME. For temporal feature learning, one-dimensional (1D) convolution is suitable for such a task. In our work, we insert densely connected 1D convolution layers inside a group of sub-convolutions and arrange it in a hierarchical layout to model the temporal representation and long-range temporal dependencies.

The main contributions of our proposed approach are as follows: (1) we design three submodules, including ME, MvE, and DCTA, to learn enriched spatiotemporal and motion representations in a very efficient way and an end-to-end manner; (2) we propose META and insert it in 2D ResNet-50 with a few additional model parameters and low computation cost; and (3) we conduct extensive experiments on three popular benchmarking datasets, including Something-something v1, Jester, and Moments in Time Mini and the results show the superiority of our approach.

The rest of the chapter is structured as follows: **Section 4.2** lists several related works that influence our proposed method. **Section 4.3** describes in detail our proposed method for the multi-class action recognition problem and addresses some limitations as discussed in **Chapter 3**. **Section 4.4** includes our extensive experiments, evaluation, comparison with the other related work, and in-depth discussion. The last section is **Section 4.5** to conclude the chapter, including a summary of our proposed approach and some future directions to enhance the performance of our approach.

4.2 Related Work

This section will discuss several related works for action recognition, including MV-CNN [49] and TEA [52], which highly motivated this work. We also add a Transformers-based approach in this section because it has contributed to recent advances in computer vision, particularly for action recognition tasks.

4.2.1 Switching from 3D to 2D CNN

With publicly available large-scale video datasets for action recognition, more 3D CNNs are introduced for action recognition tasks. As discussed earlier in 2, 3D CNN is not suitable for running in a real-world application since its computations are heavy and require more in-device memory since its parameter also contains a temporal axis alongside the spatial axis. To overcome this problem, various researchers have started exploring the use of 2D CNN for action recognition and studying how to learn a video's temporal aspect since the 2D kernel does not convolve that additional dimension.

A straightforward way is to utilize a naive 2D CNN and perform neural network learning with a two-stream configuration for action recognition tasks. Two-stream CNNs are designed to convert CNNs intended for image representation to action recognition via action decomposition and temporal information fusion. The authors' initial assumption that actions can be divided into spatial objects and temporal motion is based on the fact that actions are 3D. Following this process, spatial stream CNNs and temporal stream CNNs—each learned from action single frames and optical flow stacks across an action sequence—are added. Taking two works by Karpathy *et al.* [43] and Simonyan *et al.* [70] as examples, they introduced a two-stream CNN for RGB input and optical flow input. They considered a stream that takes the optical flow modality as the temporal stream because it is responsible for learning continuous data. However, we argue that it is inaccurate to refer to the flow stream as the temporal stream because the optical flow only represents the motion features between the neighboring frames. The structure of this stream is almost the same as the spatial stream with 2D CNN. Therefore, this flow stream cannot capture the long-range temporal relationship. Besides, extracting optical flow is expensive in both time and space, limiting vast industrial applications in the real world. Another work by Wang *et al.* [86] extracted averaged features from stridden sampled frames.

Meanwhile, learning from the three anatomical planes of the human body, i.e., sagittal, coronal, and transverse, the DEEP HRI team [49] tried to simulate the 3D convolution in their work (illustrated in Figure 4.1). A video clip can be represented as a $T \times H \times W$ volumetric data, with T, H, and W denoting the number of frames, height, and width, respectively. They separately reshape it to 2D data (i.e., $T \times H$, $T \times W$, and $H \times W$) and operate a shared 2D kernel to convolve over reshaped 2D data. Going beyond the previous work, SENet[33] also attracts researchers to employ its squeeze-and-excitation method. This method used a global context pooling mechanism to enhance the spatially informative channels and was verified to be effective in image understanding tasks. Another work by Zhou *et al.* [100] proposed Temporal Relational Network (TRN) to improve the performance of temporal augmentation by performing sampling at different temporal scales and substitute pooling operation with a fully connected network, which should encode the temporal ordering of frames.

Afterward, a work from [77] proposed R(2+1)D, a configuration of decomposed spatial and temporal modeling into two separate steps and later was adopted by [73, 16, 30, 63, 62, 74]. This strategy was used to address the heavy computation and quadratic growth of model size when using 3D convolution. Another advantage is that forcing the 3D convolution into separate spatial and temporal components makes optimization easier. This strategy is manifested in lower training errors than in 3D convolutional networks of the same capacity. However, such a decomposing strategy will double the number of nonlinearities in the network due to the additional

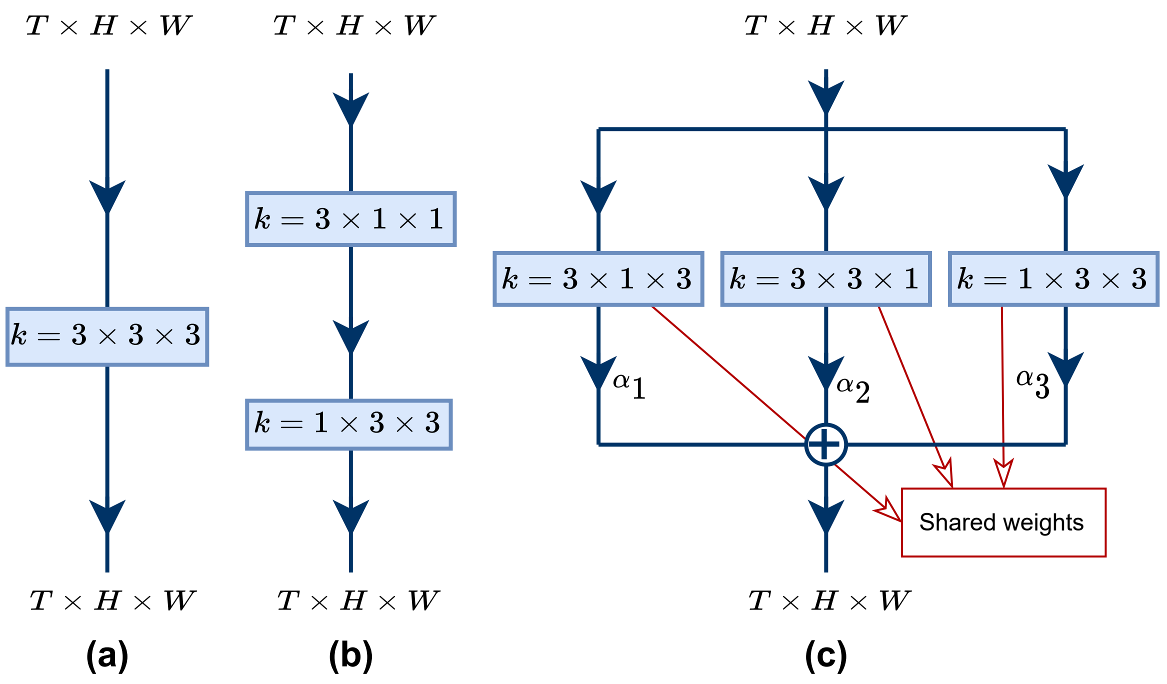


Fig. 4.1 Comparison of three designs of spatiotemporal feature learning. Assuming k is the kernel with the size of 3: (a) vanilla 3D convolution, kernel convolves on temporal (T) and spatial ($H \times W$) axes simultaneously; (b) (1+2)D convolution, temporal and spatial feature learnings are constructed serially; (c) a multi-view design from [49], multi-view feature learnings occur independently, and the resulting feature maps are aggregated with weighted average α_i to produce multi-viewed feature maps.

ReLU between the 2D and 1D convolution in each block. Increasing the number of nonlinearities increases the complexity of functions that can be represented. Illustration of R(2+1)D and comparison with other strategies to capture spatial and temporal is presented in Figure 4.1. A mix of 2D and 3D CNNs in a unified architecture is also used to capture spatial and temporal features at the same time [91]. ECO [101] utilized this mixing strategy with top-heavy architecture. However, mixing the 2D CNN with the 3D CNN in one architecture will inevitably increase model parameters and require more time to optimize the parameters.

4.2.2 Squeeze and Excitation Mechanism

The central building block of CNNs is the convolution operator, which enables networks to construct informative features by fusing spatial and channel-wise information within local receptive fields at each layer. A broad range of prior research has investigated the spatial component of this relationship, seeking to strengthen the representational power of a CNN by enhancing the quality of spatial encodings throughout its feature hierarchy. In this work, they focus instead on the channel relationship and propose a novel architectural unit that adaptively recalibrates channel-wise feature responses by explicitly modeling interdependencies between channels. They further demonstrate that the proposed blocks bring significant improvements in performance for existing state-of-the-art CNNs at a little additional computational cost.

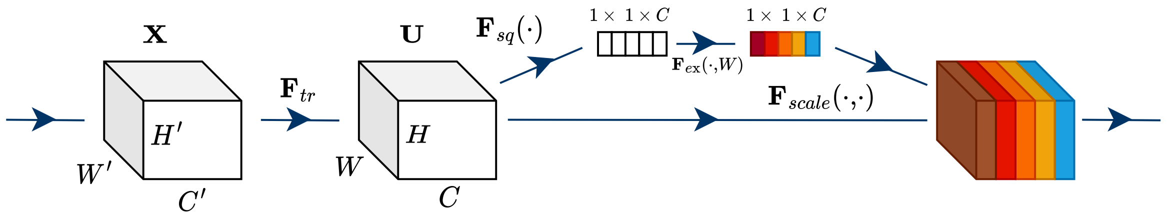


Fig. 4.2 Feature recalibration on channel responses by squeezing feature maps across spatial dimensions and performing a self-gating mechanism to excite the embedding.

4.2.3 Motion Modeling

Besides spatiotemporal representation, motion information is also the key difference between action and image classification tasks; thus, exploiting motion features is mandatory. A comprehensive study by Sevilla-Lara *et al.* [68] analyzed the importance of optical flow for action recognition. They analyzed the optical flow itself, conducted several experiments using optical flow modality, and proved the contribution to the accuracy. Another several attempts utilized optical flow as an additional input to RGB [70, 92, 86, 1]. While this modality demonstrates excellent results compared to RGB data alone, this approach cannot be implemented in a real-world application as extracting the pixel-wise optical frame with the TV-L1 method [97] requires heavy computation as well as much storage space.

The RGB difference was introduced in this light, which is more lightweight. A breakthrough innovation was introduced by Jiang *et al.* [41]. They proposed an STM module that first outlined the motion calculation for end-to-end learning in a 2D ConvNet and proved effective in capturing instantaneous motion representation as short-range temporal evolution. The proposed work is illustrated in Figure 4.3. Furthermore, Li *et al.* [52] introduced a new block, termed TEA, to explore the benefits of the attention mechanism added to the previously mentioned motion calculation. In addition, they suggested overcoming the limitation of long-range temporal representation by introducing multiple temporal aggregations in a hierarchical design. In this work, we propose a motion calculation and a hierarchical structure of local temporal convolutions, similar to the previous work. We will explain more details of our work and highlight the difference from the previous work in the subsequent section.

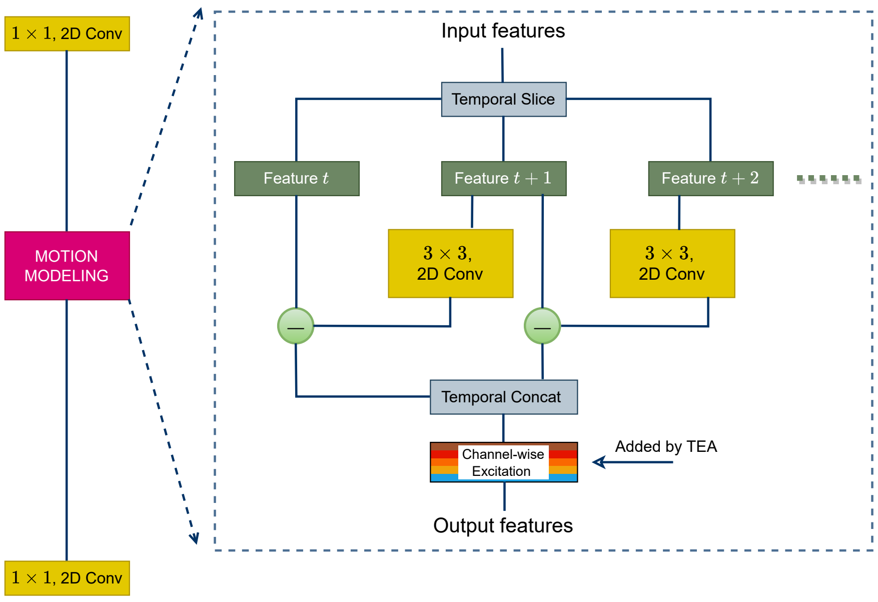


Fig. 4.3 RGB difference at feature-level.

4.2.4 Temporal Shifting Mechanism

Temporal Shift Modules have proposed another approach for capturing temporal contexts (TSM) [53]. This approach shifts part of the channels along the temporal dimension forward and backward, allowing information to be exchanged among neighboring frames.

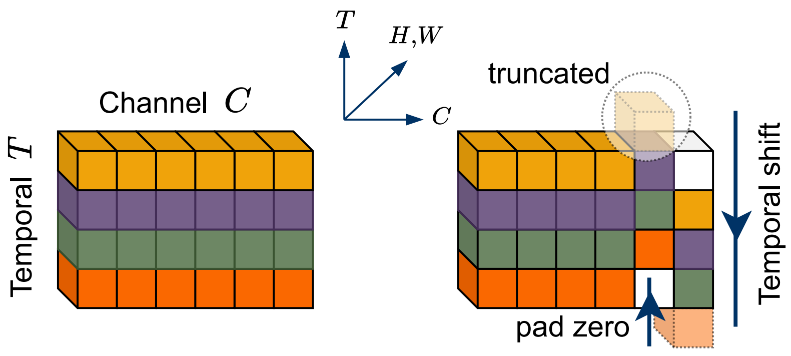


Fig. 4.4 Comparison between tensor without and with part shift mechanism: (a) the tensor without shift mechanism; (b) the tensor with shift mechanism, only part of the tensor is shifted to give an opportunity of the model to learn temporal dimension.

As illustrated in 4.4, the information from neighboring frames is mingled with the current frame after shifting. They argued that the convolution operation consists of shift and multiply-accumulate. They shift in the time dimension by ± 1 and fold the multiply-accumulate from the time dimension to the channel dimension.

4.2.5 Vision Transformer

As Vision Transformers brought recent breakthroughs in computer vision, specifically for action recognition tasks, many researchers have adopted it as their model [4, 5, 78, 50] or combined it with 2D CNN [75]. For example, Arnab *et al.* [4] proposed several factorization variants to model spatial and temporal effectively inside the transformer encoder, as illustrated in Figure 4.5. TimeSformer [5] investigated several self-attention combinations on frame-level patches and suggested that separated attention for spatial and temporal applied within each block yielded the best video classification accuracy.

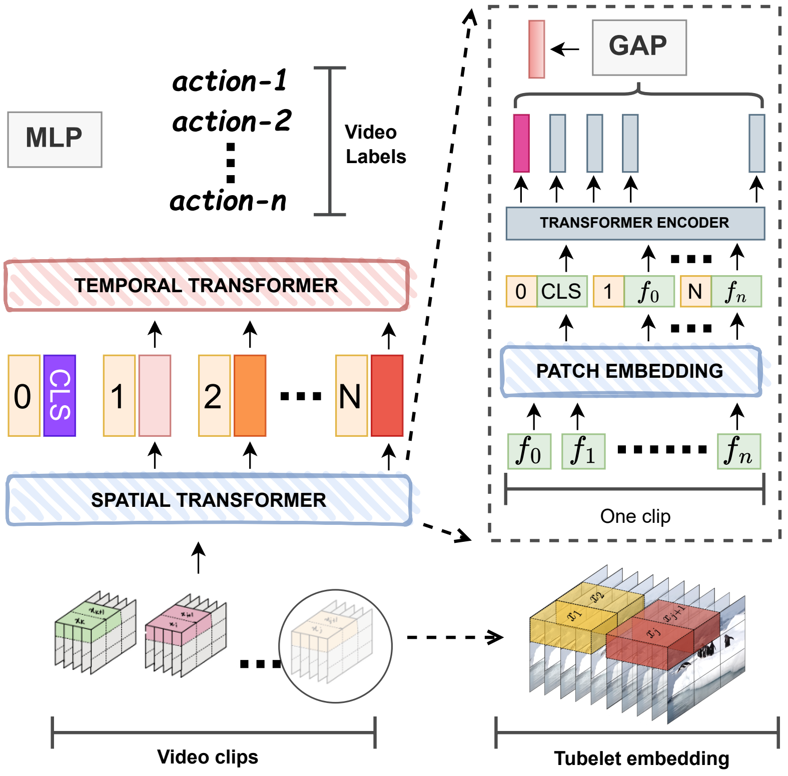


Fig. 4.5 Each clip is passed through the spatial transformer and results in an encoding vector for each clip with CLS token arranged in appropriate position embedding. This positioned embedding is then passed through a temporal transformer encoder. Each token is a vector extracted from a clip for the temporal transformer, so each token is from a different temporal index in the video.

Another work by Tian *et al.* [75] introduced a 2D ResNet with Transformer injected at the top layer before the Linear layer to accurately aggregate extracted local cues from preceding blocks into a video representation. Although these current approaches seem promising, a Transformer-based network is not suited for real-world applications because it is highly compute-intensive [94].

4.3 Proposed Excitation and Aggregation Networks

This section will discuss the technical details of our work. Firstly, we will present a method to extract motion representations to simulate the optical flow modality. Afterward, our novel multi-view excitation is introduced. Lastly, a simple stacking local temporal convolution with dense connection is also discussed here as a part of our improvement strategy. Some notations written in this section are: N, T, C, H, and W indicate the batch size, the number of frames, the number of channels, height, and width, respectively.

4.3.1 Motion Excitation (ME)

Introduced firstly by STM[41] and later enhanced by TEA[52], the motion excitation submodule performs frame difference calculation in a unified framework for end-to-end learning. In principle, motion representation indicates content displacement of two adjacent feature maps, therefore called *feature-level* based motion, rather than *pixel-level* based motion, as in the concept of optical flow. Fig. 4.7 illustrates steps to measure approximate feature-level temporal differences.

First step is to reduce the number of channels for efficiency with ratio $r = 16$ by applying 1×1 convolution layer K_{red} to the initial input X, formulated in (4.1). Then, we slice feature maps at the temporal axis, followed by element-wise subtraction for every adjacent output feature, and obtain M at time step t. Before subtraction, a 3×3 transformation convolution layer K_{transf} is applied to the output features X' at the time step $(t + 1)$. Next, we concatenate motion representations M at all time steps according to the temporal axis with 0 padded to the last segment. Concretely, given $X \in \mathbb{R}^{NT \times C \times H \times W}$ is input features to the ME submodule, the above processes

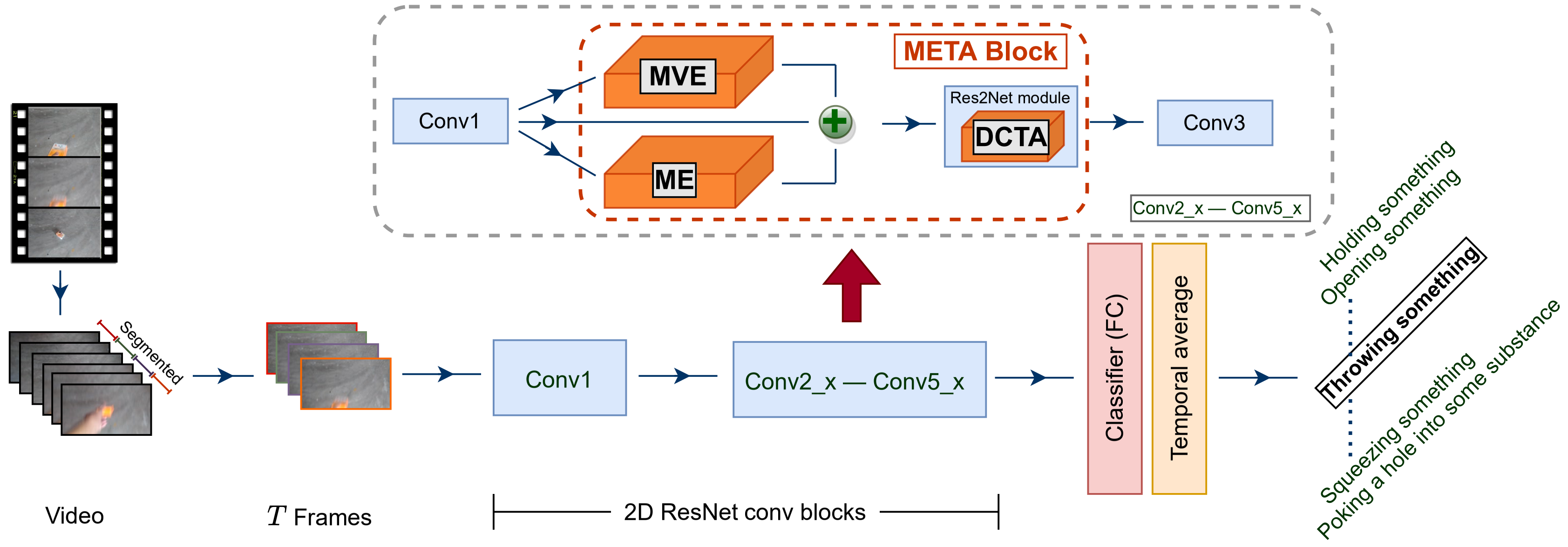


Fig. 4.6 An overview of our proposed model implemented in 2D ResNet50 [31] architecture. We replace the original "conv2" with ME, MvE, and DCTA inside every residual block to construct the META block. Inside Res2Net [21] module, we insert a DCTA submodule. Details on data flow are given in Section 4.4.2.

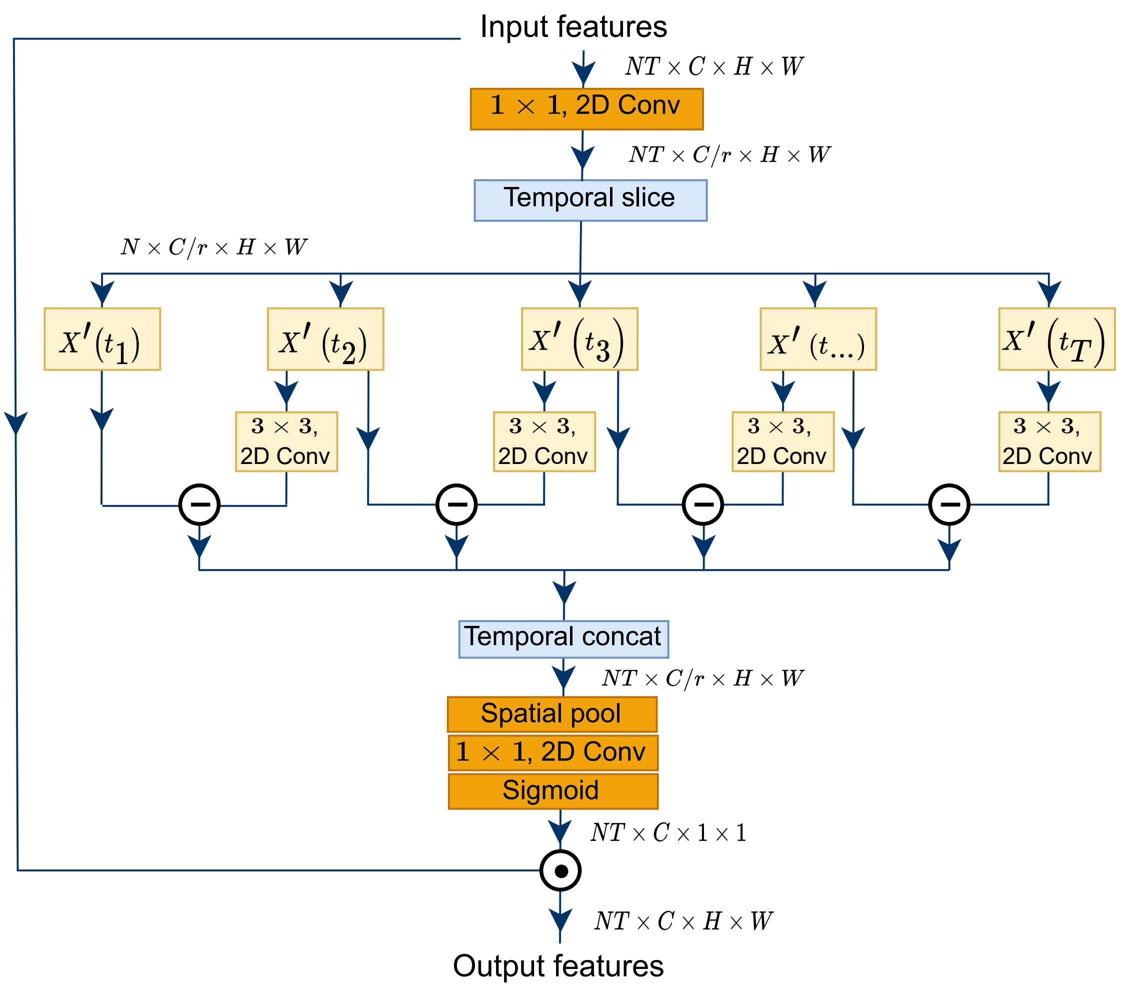


Fig. 4.7 Two adjacent feature maps are subtracted to obtain motion representation. We firstly apply channel-wise 3 × 3 convolution on frames $[t+1]$ before subtraction.

are expressed as follows:

$$X' = K_{red} * X, \quad X' \in \mathbb{R}^{NT \times \frac{C}{r} \times H \times W} \tag{4.1}$$

$$M_t = K_{transf} * X'_{t+1} - X'_t, \quad 1 \geq t \geq T-1 \tag{4.2}$$

$$X' = concat(M_t, 0), \quad 1 \geq t \geq T-1 \tag{4.3}$$

where $M_t \in \mathbb{R}^{N \times \frac{C}{r} \times H \times W}$ and last $X' \in \mathbb{R}^{NT \times \frac{C}{r} \times H \times W}$.

Figure 4.8 illustrates the process of adding all zeros to the last temporal order. This procedure exists to ensure that the number's temporal size remains unchanged after subtraction.

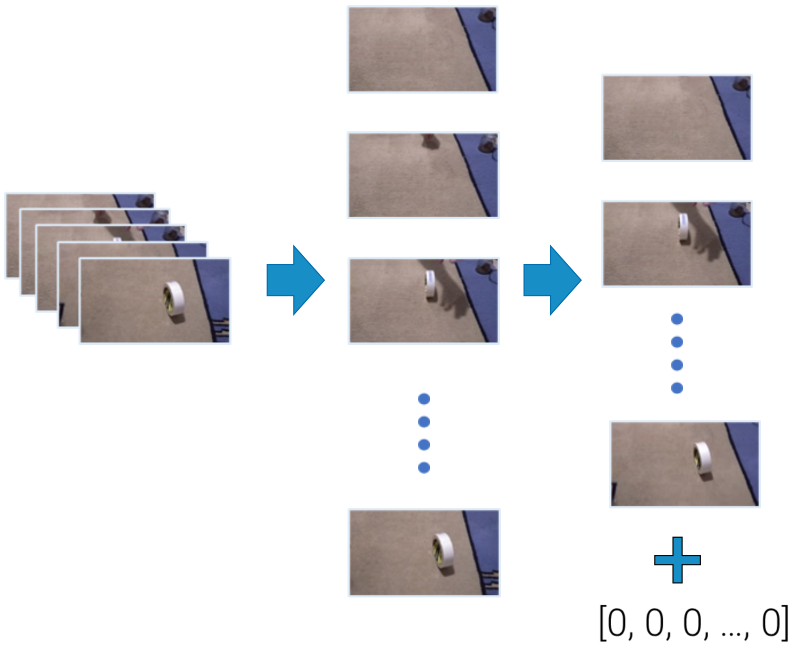


Fig. 4.8 An illustration of zero padding. We can devide this illustration into three columns. The first column is the input feature maps. The second column is the splitted feature maps followed by subtraction process. The third column is the motion representation at timestep t. We add zeros to the last timestep.

At this point, we have new X' as approximate feature-level motion representations. Since we want to emphasize the informative features and suppress less useful ones alongside with [33], we squeeze the global information from each channel of the motion representations by utilizing the global spatial pooling layer. Then, another 1×1 2D convolution layer K_{exp} performs channel expansion to restore the number of channels and we obtain new X, as in (4.4). Lastly, attentive feature maps are obtained by feeding new X to a sigmoid function δ while final outputs X_{ME} are produced from a multiplication between the initial inputs X and attentive feature maps, as defined by (4.5).

$$X = K_{exp} * pool(X'), X \in \mathbb{R}^{NT \times C \times 1 \times 1} \tag{4.4}$$

$$X_{ME} = \delta(F) * X, \quad X_{ME} \in \mathbb{R}^{NT \times C \times H \times W} \tag{4.5}$$

When subtracting the feature maps, we only calculate it one time: a collection of feature maps containing $[2 \sim T]$ timestamps minus $[1 \sim T-1]$. For more detailed flow of the proposed framework, we present in pseudo-code Algorithm 1.

Algorithm 1: Motion estimation in the Motion Excitation submodule

```
Input      : 4D tensor $X$ with shape of $NT \times C \times H \times W$
Parameters : $N$ the batch size, $T$ the temporal number, $C$ the channel
             number, both $H$ and $W$ are the spatial size, $C_{red}$ the reduced
             channel number
Output     : 4D tensor $X'$ with shape of $NT \times C \times H \times W$
1  $X_{red}$ = 2DConvReduction ($X$) /* channel reduction */
2  $X_{red} \leftarrow$ TensorView ($X_{red}$) /* $N \times T \times C_{red} \times H \times W$ */
3  $X^0_{red} \leftarrow$ TensorSplitForT ($X_{red}$) /* to obtain $[1 \sim T-1]$ tensors */
4  $X^1_{red} \leftarrow$ TensorSplitForT+1 ($X_{red}$)/* to obtain $[2 \sim T]$ tensors */
5  $X^1_{red} \leftarrow$ 2DConv ($X^1_{red}$)
6  $X_{red} \leftarrow X^1_{red}$ - $X^0_{red}$ $X_{red} \leftarrow$ Padding ($X_{red}$)
   #Expand the tensor and perform excitation
7  $X'_{red} \leftarrow$ Pool ($X_{red}$) /* $NT \times C_{red} \times 1 \times 1$ */
8  $X' \leftarrow$ 2DConvExpand ($X'_{red}$) /* $NT \times C \times 1 \times 1$ */
9  $X' \leftarrow$ Sigmoid ($X'$) /* $NT \times C \times 1 \times 1$ */
10 $X' \leftarrow X \times X'$ /* $NT \times C \times H \times W$ */
```

4.3.2 Densely Connected Temporal Aggregation (DCTA)

Previously, learning temporal relationships in the task of action recognition is achieved by repeatedly stacking local temporal layers in deep networks. Unfortunately, it will raise some problems. It is considered to be harmful to the features because the optimization message transmitted from distant frames has been weakened. To alleviate such a problem, we propose the Densely Connected Temporal Aggregation submodule. We follow Res2Net design[21] to split feature maps in channel dimension into four subgroups of convolutions separately. Each subgroup consists of temporal and spatial convolutions configured serially, while one subgroup has temporal convolution only. In addition, output features from each subgroup flow to the next convolutional block and the neighboring subgroup through a residual connection, except for one subgroup without a residual-like connection (see DCTA submodule in Fig. 4.9 for the detail). Thus, the last subgroup *aggregately* receives refined spatiotemporal features from former subgroups.

Regarding temporal convolution, we arrange the layers in a stacked and *densely connected* fashion. Notably, its parameters are shared across subgroups. In this work, the number of temporal convolution layers for stacking is three and these stacked layers are placed in three subgroups having a residual connection. More specifically, the first layer receives the encoded features from the summation of ME and MvE; the second

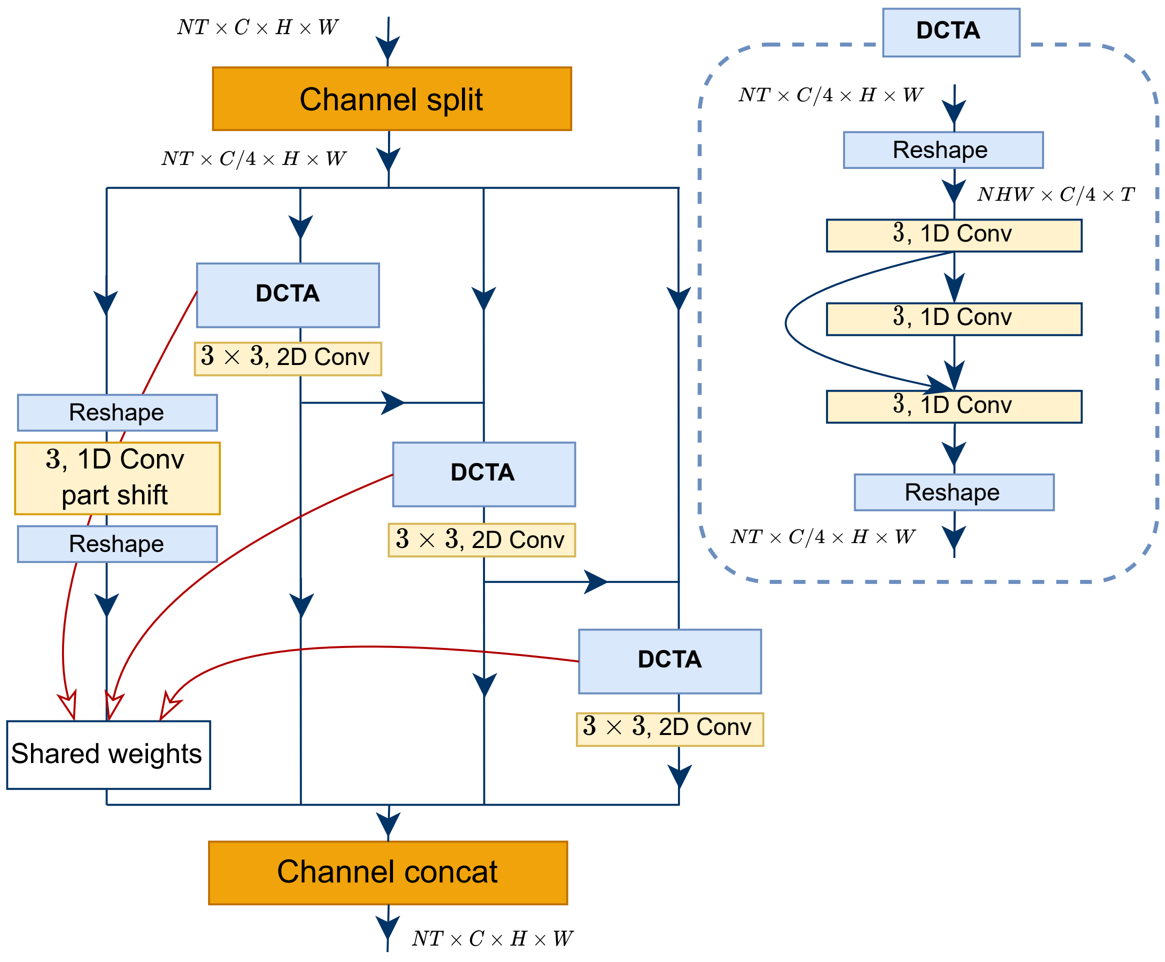

$NT \times C \times H \times W$
Channel split
$NT \times C/4 \times H \times W$
DCTA
3 × 3, 2D Conv
Reshape
3, 1D Conv part shift
Reshape
DCTA
3 × 3, 2D Conv
DCTA
3 × 3, 2D Conv
Shared weights
Channel concat
$NT \times C \times H \times W$
DCTA
$NT \times C/4 \times H \times W$
Reshape
$NHW \times C/4 \times T$
3, 1D Conv
3, 1D Conv
3, 1D Conv
Reshape
$NT \times C/4 \times H \times W$

Fig. 4.9 A detailed diagram showing the architecture of the DCTA submodule inside Res2Net module.

layer receives input features from the first layer; for the third layer, its input is formed from the summation of all the preceding layers' output features. Formally,

$$\left.\begin{aligned} X'_0 &= K_{temp} * X \\ X'_1 &= K_{temp} * X'_0 \\ X'_T &= K_{temp} * (X'_0 + X'_1) \end{aligned}\right\} \tag{4.6}$$

where K_{temp}, X, $X'_i \in \mathbb{R}^{NHW \times C \times T}$, $X'_T \in \mathbb{R}^{NHW \times C \times T}$ denote 1D convolution with a kernel size of 3, initial input features, output features from the i–th layer, and the final result of the last temporal layer. We omit necessary permutation and reshape for X and X'_T for simplicity. After that, a 3×3 spatial convolution follows, as stated in the previous paragraph. For all subgroups in DCTA submodules, the process can mathematically be expressed as:

$$\left.\begin{aligned} X'_0 &= K_{temp} * X \\ X'_1 &= K_{spa} * X'_T \\ X'_2 &= K_{spa} * (X'_1 + X'_T) \\ X'_3 &= K_{spa} * (X'_2 + X'_T) \end{aligned}\right\} \tag{4.7}$$

where $X'_i \in \mathbb{R}^{NT \times \frac{C}{4} \times H \times W}$, K_{spa}, and K_{temp} are output features of the i–th subgroup, spatial convolution, and part-shift temporal convolution from[53], respectively. Lastly, we concatenate across channel dimensions to obtain the final output features X':

$$X' = concat([X'_i]), \quad i = [0, 1, 2, 3] \tag{4.8}$$

where $X' \in \mathbb{R}^{NT \times C \times H \times W}$. Notice that in Fig. 4.9, indices of subgroups from left to right are from 0 to 3, correspondingly.

4.3.3 Multi-view Excitation (MvE)

Learning spatiotemporal features is crucial for action recognition in videos. Current models of deep neural networks can independently learn spatial and temporal characteristics, similar to a C2D design. Different from previous known approaches, we try to decompose the spatiotemporal information into seperate spatial and temporal signal, as can be seen in Figure 4.10. Vision of H×W is the natural perspective with which humans are acquainted while snapshots from perspectives involving T (i.e. T×W and T×H) are difficult for humans to interpret. Although images in the latter views are difficult to interpret, they contain exactly the same amount of information as the normal H×W view.

As illustrated in Fig. 4.11, the MvE submodule has three branches to extract beneficial information from different views, similar to that [49]. Given an input feature $X \in \mathbb{R}^{NT \times C \times H \times W}$ for branch TH, we utilize 1×1 2D convolution layer K_{red} to reduce the channel number for efficiency with ratio $r = 16$, identical to (4.1). Then, tensor dimension is reshaped to comply with desired dimension, i.e., $NT \times \frac{C}{r} \times H \times W \to NW \times \frac{C}{r} \times T \times H$. After that, a shared *channel-wise* convolution layer K is utilized to produce transformed feature maps X'_{TH}. Formally,

$$X'_{TH} = K * X', \quad X'_{TH} \in \mathbb{R}^{NW \times \frac{C}{r} \times T \times H} \tag{4.9}$$

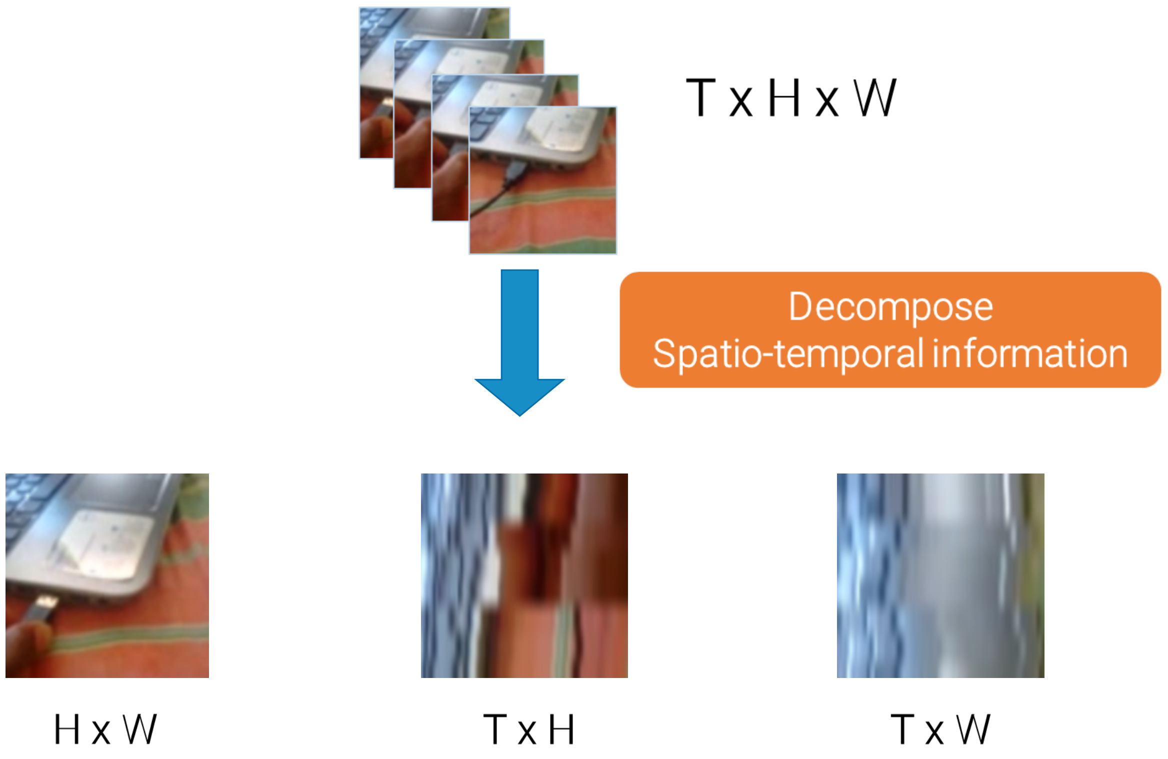


Fig. 4.10 Visualization of decomposed spatiotemporal images in different views. T, H, and W denote temporal size, height, and width, respectively.

Last step is to reshape back the tensor dimension, i.e., $NW \times \frac{C}{r} \times T \times H \rightarrow NT \times \frac{C}{r} \times H \times W$. The rest of the branches are processed accordingly to produce X'_{TW} and X'_{HW}. If we have obtained all the outputs from the other two branches, then new X' is a convex combination of the X'_i:

$$X' = \sum_i \alpha_i * X'_i, \quad i \in [TH, TW, HW] \tag{4.10}$$

where α is a weighted average or coefficient with constraints of $\sum_i \alpha_i = 1$ and each of the $\alpha_i \geq 0$. We argue that each branch will contribute differently to the performance of the model. For the rest of operations are identical to (4.4) and (4.5) to obtain attentive multi-view feature maps X_{MvE}. For more detailed flow of the proposed framework, we present in pseudo-code Algorithm 2. By splitting the input feature maps into three different views like this way, the "cross-slice" information extraction may be achieved while keeping the computation cost at a low level.

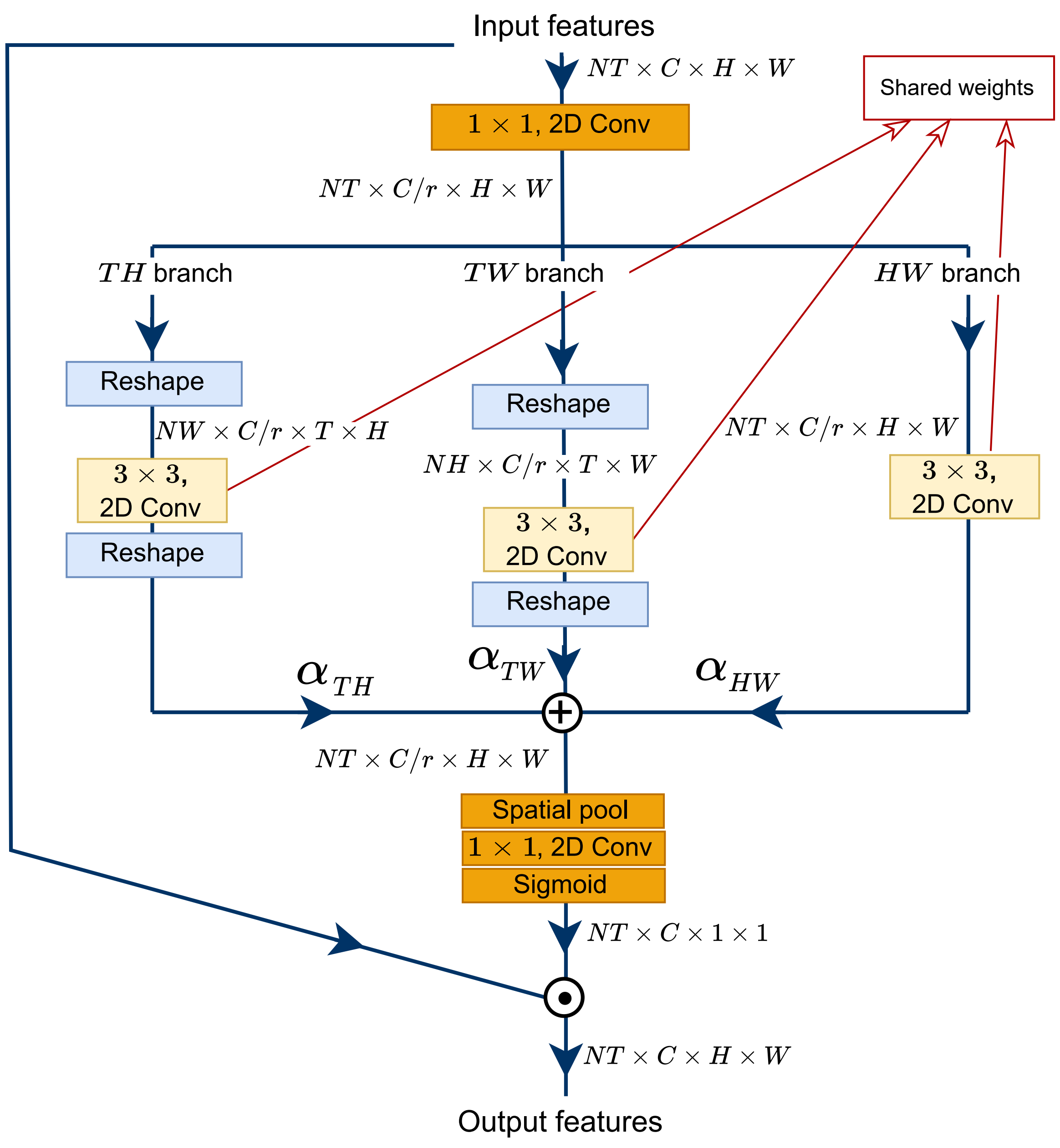


Fig. 4.11 Detailed architecture of MvE submodule.

Initial work of multi-view design was proposed by Li *et al.* with their team DEEP HRI[49]. Different from their work, our work introduces the excitation algorithm to the MvE submodule so that it has a kind of attention mechanism.

Algorithm 2: Spatial and temporal learning architecture in multi-vew excitation submodule

```
Input      : 4D tensor $X$ with shape of $NT \times C \times H \times W$
Parameters : $N$ the batch size, $T$ the temporal number, $C$ the channel
             number, both $H$ and $W$ are the spatial size, $C_{red}$ the reduced
             channel number, $\alpha$ the learned coefficient
Output     : 4D tensor $X'$ with shape of $NT \times C \times H \times W$

1  $X_{red}$ = 2DConvReduction $(X)$ /* channel reduction */
   #Learning for T-H branch
2  $X_{red}^{TH} \leftarrow$ TensorView $(X_{red})$ /* $N \times T \times C_{red} \times H \times W$ */
3  $X_{red}^{TH} \leftarrow$ TensorPermute $(X_{red}^{TH})$ /* $N \times W \times C_{red} \times T \times H$ */
4  $X_{red}^{TH} \leftarrow$ TensorView $(X_{red}^{TH})$ /* $NW \times C_{red} \times T \times H$ */
5  $X_{red}^{TH} \leftarrow$ 2DConvShared $(X_{red}^{TH})$ /* perform 2D convolution */
6  $X_{red}^{TH} \leftarrow$ TensorView $(X_{red}^{TH})$ /* $N \times W \times C_{red} \times T \times H$ */
7  $X_{red}^{TH} \leftarrow$ TensorPermute $(X_{red}^{TH})$ /* $N \times T \times C_{red} \times H \times W$ */
8  $X_{red}^{TH} \leftarrow$ TensorView $(X_{red}^{TH})$ /* $NT \times C_{red} \times H \times W$ */
   #Learning for T-W branch
9  $X_{red}^{TW} \leftarrow$ TensorView $(X_{red})$ /* $N \times T \times C_{red} \times H \times W$ */
10 $X_{red}^{TW} \leftarrow$ TensorPermute $(X_{red}^{TW})$ /* $N \times H \times C_{red} \times T \times W$ */
11 $X_{red}^{TW} \leftarrow$ TensorView $(X_{red}^{TW})$ /* $NH \times C_{red} \times T \times W$ */
12 $X_{red}^{TW} \leftarrow$ 2DConvShared $(X_{red}^{TW})$ /* perform 2D convolution */
13 $X_{red}^{TW} \leftarrow$ TensorView $(X_{red}^{TW})$ /* $N \times H \times C_{red} \times T \times W$ */
14 $X_{red}^{TW} \leftarrow$ TensorPermute $(X_{red}^{TW})$ /* $N \times T \times C_{red} \times H \times W$ */
15 $X_{red}^{TW} \leftarrow$ TensorView $(X_{red}^{TW})$ /* $NT \times C_{red} \times H \times W$ */
   #Learning for H-W branch
16 $X_{red}^{HW} \leftarrow$ 2DConvShared $(X_{red})$ /* perform 2D convolution */
   #Weighted summation with $\alpha$
17 $X_{red}^{ALL} \leftarrow \alpha_{TH} \times X_{red}^{TH} + \alpha_{TW} \times X_{red}^{TW} + \alpha_{HW} \times X_{red}^{HW}$
   #Expand the tensor and perform excitation
18 $X'_{red} \leftarrow$ Pool $(X_{red}^{ALL})$ /* $NT \times C_{red} \times 1 \times 1$ */
19 $X' \leftarrow$ 2DConvExpand $(X'_{red})$ /* $NT \times C \times 1 \times 1$ */
20 $X' \leftarrow$ Sigmoid $(X')$ /* $NT \times C \times 1 \times 1$ */
21 $X' \leftarrow X \times X'$ /* $NT \times C \times H \times W$ */
```

4.3.4 META Block

For comparative purposes, we adopt 2D ResNet-50 as a backbone like other state-of-the-art methods [98, 88, 52]. As shown in Figure 4.6, each "conv2" in all residual blocks (conv2_x until conv5_x) is replaced by META. In total, we insert 16 blocks of META to endow the network with the ability to learn both spatiotemporal and motion representations efficiently. When feeding feature maps to the DCTA submodule, we sum all of the output features generated from ME, MvE, and former convolution block (denoted by X_{ME}, X_{MvE}, and X, respectively) to obtain X'.

$$X' = X_{ME} + X_{MvE} + X, \quad X' \in \mathbb{R}^{NT \times C \times H \times W} \tag{4.11}$$

4.3.5 Discussion with TEA

We want to highlight the differences between our work and TEA in this subsection. TEA adopted feature-level motion representation and enhanced it by excitation strategy with negligible extra model parameters. Unlike TEA, where it only considers X in parallel with the output of ME, we have also added output features from the MvE submodule, as in Equation (4.11). Moreover, the network enjoys richer spatiotemporal and motion representation features since we re-calibrate the features by *both* ME and MvE submodules.

Regarding temporal aggregation inside the Res2Net module, TEA adopted it to enable their network to model the long-range spatiotemporal relationship by adding a local temporal convolution to each sub-group of convolution. While in our work, we also added local temporal convolutions in each sub-group of convolution and arranged it in *stacked up* and *densely connected* manner.

4.4 Experiments and Evaluation

4.4.1 Dataset

Our proposed method is evaluated on three large-scale action recognition benchmark datasets, i.e., Something-something v1, Jester, and Moments in Time Mini.

An action classification on the Something-something v1 [23], a motion-centric type of dataset with 174 classes, requires temporal understanding to classify an action. This dataset is designed to emphasize the interaction between human and object, for example, "*Throwing something*" and "*Throwing something in the air and catching it*". It contains

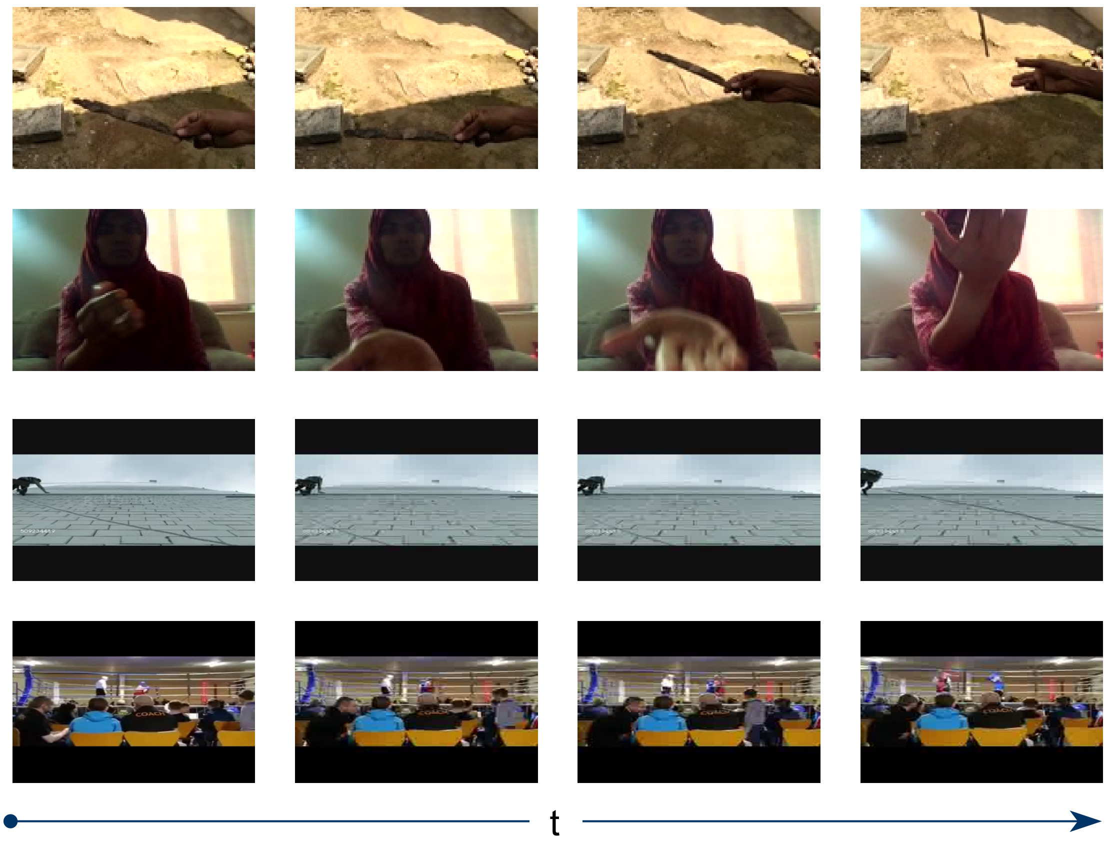


Fig. 4.12 Examples of frames from (top-down) Something-something v1 ("*Throwing something*"), Jester ("*Swiping up*"), and Moments in Time Mini ("*repairing*", "*boxing*") datasets.

108,499 videos, with 86,017 in the training set and 11,522 in the validation set. Jester [55], which is also considered a temporal-related dataset, consists of 118,562 training videos, 14,787 validation videos, and fewer categories than the Something-something v1 dataset, i.e., 27. Example actions are "*Swiping up*" and "*Zooming out with two fingers*". Moments in Time Mini dataset [57] is a large-scale human annotated collection of one hundred thousand short videos corresponding to dynamic events unfolding within three seconds. "*boxing*" and "*repairing*" are the two examples of categories. This dataset provides 100,000 videos for training and 10,000 for validation. It involves 200 action categories and offers a balanced number of videos in each category. While the previous datasets are more temporal-related datasets, the Moments in Time Mini dataset can be considered both a temporal and scene-related dataset. Frames have already been extracted from all videos in the Something-something v1 and Jester datasets when they are publicly available. However, in the Moments in Time Mini dataset, we must extract RGB frames from the videos at 30 frames per second at a resolution of 256 by 256. Figure 4.12 shows some images with its class for aforementioned datasets.

4.4.2 Experimental Setup

Training. We conduct all experiments on one Nvidia Quadro P6000 GPU card with PyTorch [58] as the deep learning framework. We follow a sparse sampling strategy by TSN [86]. We extract T frames randomly from a number of segments (in all our experiments, $T = 8$). Selected frames go through the network and simple temporal pooling strategy is utilized to averagely predict an action for an entire video. Random scaling and cropping are applied as data augmentation. The size of the shorter side of the frame is cropped to 256. The cropped region will be resized to 224×224 and serve as the final frame size; hence the final input shape is $N \times T \times 3 \times 224 \times 224$. Before the training started, we loaded our base model with weights trained on the popular ImageNet dataset [14, 67]. As we adopt Res2Net module for residing the DCTA submodule, we select the publicly available *res2net50 26w 4s*[1] pre-trained weights.

Testing. We follow settings from[52], to adopt two methods as testing protocols: (1) efficient protocol, with frames × crops × clips is 8 × 1 × 1 and cropped 224 × 224 at central region as final frame size; (2) accuracy protocol, with frames × crops × clips is 8 × 3 × 10, full resolution images (256 × 256 for final input size for frames) and averaged softmax scores for all clips (in our work, 10 clips) for final prediction. When comparing with other recent works, we apply the accuracy protocol, as in Table 4.3 and Table 4.4. For the Moments in Time Mini dataset, as in Table 4.5, we apply the efficient protocol.

4.4.3 Evaluation Measure

In action understanding, multi-class classification consists of problems where the model returns per-class confidence scores for each input video. This is done primarily with a softmax loss in which the confidence scores across classes for a given input sum to one. We use top-k categorical accuracy as our metric. This metric measures the proportion of times when the ground truth label can be found in the top k predicted classes for that input. Top-1 accuracy, sometimes simply referred to as accuracy, is the most ubiquitous while top-3 and top-5 are other standard choices. To calculate Top-k accuracy, let $\hat{y}_k^{(i)} \subseteq \hat{y}^{(i)}$ be the subset containing the k highest confidence scores for

[1]available at https://shanghuagao.oss-cn-beijing.aliyuncs.com/res2net/res2net50_26w_4s-06e79181.pth

video $x_{(i)}$. The top-k accuracy (a_k) over the entire input set, where 1 is a 0-1 indicator function, is defined as:

$$a_k = \frac{1}{n} \sum_{i=1}^{n} 1_{\hat{y}_k^{(i)}}(y^{(i)}) \tag{4.12}$$

4.4.4 Result & Analysis

In this subsection, we explain the result investigation according to the ablation study to know the impacts from specific system parts. All ablative experiments were conducted using the same settings, i.e., the efficient protocol. We load the network with weights trained on the ImageNet dataset. To demonstrate and examine the effect of each component of the building block on its performance, we conducted investigations into two parts:

- **Impact of each submodules.** We examine how each submodule affects the performance and present the findings in Table 4.1. It is clear that, in comparison to the baseline, each submodule continuously improves the performance of the 2D ResNet on video action recognition. The DCTA submodule makes the most contribution, improving top-1 accuracy by 2.4% while being computationally efficient with only a 1.7G overhead gap and the least number of parameters, whereas the other two add 2.0G of extra FLOPs.

- **Location of META. Location of META.** We examine the number of META implemented inside four convolution blocks toward accuracy. From Table 4.2, it is evident that better precision can be attained with more profound METAs placed in convolution blocks. Interestingly, META only requires installing one convolution block to dramatically increase the performance, with top-1 and top-5 accuracies exceeding the baseline by 2.6% and 2.9%, respectively.

Table 4.1 The comparison result of an individual component, including FLOPs and the number of parameters. We use TSM as a baseline.

Method	**FLOPs**	**# Param.**	**Top-1 %**	**Top-1 %**
TSM [52]	33.0G	23.7M	45.6	74.2
ME	35.0G	26.1M	47.9	77.8
MvE	35.0G	26.1M	46.3	76.9
DCTA	34.7G	25.7M	48.0	77.0
META	35.6G	26.6M	50.1	78.5

Table 4.2 Ablative experiments of the number and location of META to verify the contribution of META to the performance. The more META implemented, the more accuracy can be achieved.

Location	**Top-1** (%)	**Top-5** (%)	△ **Top-1** (%)	△ **Top-5** (%)
conv{2}_x	48.2	77.1	–	–
conv{2,3}_x	49.5	78.1	+1.3	+1.0
conv{2,3,4}_x	49.9	78.2	+1.7	+1.1
conv{2,3,4,5}_x	50.1	78.5	+1.9	+1.4

4.4.5 Comparison with Related Work

We report our experimental results and compare them with state-of-the-art methods. We list TSM to act as the baseline for Table 4.3 and Table 4.4. Since META is designed to function on CNN-based networks, we primarily compare our work with others whose networks are the same type as META to make relevant comparisons. Nevertheless, we still include recent Transformer-based networks in our comparison to demonstrate that META can achieve competitive accuracy while still being lightweight.

Something-something-v1. Something-something v1 can be categorized as a temporal-related dataset; thus, ME and DCTA play important roles here. We divide Table 4.3 into two compartments; the upper part is 3D-based and the rest is 2D-based CNNs. It can be seen from that table that our methods exceed the baseline method by a significant margin of 2.4% and 1.7% for Top-1 and Top-5 accuracy, respectively. Our work is superior in that even with eight frames and the one clip-one crop protocol utilized, our method still surpasses the 8+16 frames of the baseline method with slight 0.4% performance improvement. Compared to the first compartment, our accuracy protocol still gained the highest score by a large margin, a remarkable improvement of 4% (50.1% vs. 46.1%). A more competitive result of our work is shown in the second compartment, where we outperform all current state-of-the-art methods in terms of Top-1 accuracy. The nearest score to ours is TEA, where we obtain a substantially higher margin (52.1% vs. 51.7%) except Top-5 accuracy is 0.3% lower (80.2% vs. 80.5%). For comparison with ACTION-NET, a more recent work, we significantly outperform their work by a big margin of 2.9% using eight frames as input frames. This definitely demonstrates our superior submodules of MvE and DCTA combined with ME, considering ACTION-NET also equipped their network with ME.

Table 4.3 The comparison result of META against other state-of-the-art methods on the Something-something v1 dataset. RN in column backbone indicates ResNet. We list UniFormer methods with 16 input frames for relevant comparison. The highest accuracies for CNN-based networks are highlighted in bold.

Methods	**Backbone**	**Pre-train**	**Inputs**	**FLOPs**	**Param.**	**Top-1 (%)**	**Top-5 (%)**
3D CNNs:							
ECO RGB from [53]	BNInception + 3D RN-18	Kinetics	$8\times1\times1$	32.0G	47.5M	39.6	–
ECO RGB from [53]			$16\times1\times1$	64.0G	47.5M	41.4	–
I3D NL RGB [87]	3D RN-50	ImgNet + Kinetics	$32\times1\times2$	$168.0\text{G}\times2$	35.3M	44.4	76.0
I3D NL+GCN RGB [87]				$303.0\text{G}\times2$	62.2M	46.1	76.8
2D CNNs:							
TSM RGB [53]	2D RN-50	ImgNet	$8\times1\times1$	33.0G	24.3M	45.6	74.2
TSM RGB			$16\times1\times1$	65.0G	24.3M	47.2	77.1
STM [41]			$8\times3\times10$	$33.3\text{G}\times30$	24.0M	49.2	79.3
STM			$16\times3\times10$	$67.0\text{G}\times30$	24.0M	50.7	80.4
TEA [52]			$8\times1\times1$	35.1G[2]	26.1M[2]	48.9	78.1
TEA			$8\times3\times10$	$35.1\text{G}\times30$[2]	26.1M[2]	51.7	**80.5**
ACTION-NET [88]			$8\times1\times1$	34.7G	28.1M	47.2[3]	75.2[3]
MEST [98]			$8\times1\times1$	34.0G	25.7M	47.8	77.1
MEST			$16\times1\times1$	67.0G	25.7M	50.1	79.1
AIA TSM [26]			$8\times1\times1$	33.1G	23.9M	49.2	77.5
SMNet [94]			$8\times3\times10$	$33.1\text{G}\times30$	23.9M	49.8	79.6
Transformers:							
UniFormer-B [50]	Transformer	Kinetics	$16\times1\times1$	96.7G	49.7M	55.4	82.9
UniFormer-S [50]				41.8G	21.3M	53.8	81.9
EAN RGB+LMC [75]	Transformer + 2D RN-50	ImgNet	$(8\times5)\times1\times1$	37.0G	36.0M[1]	53.4	81.1
Ours:							
META	2D RN-50	ImgNet	$8\times1\times1$	35.6G	26.6M	50.1	78.5
META			$8\times3\times1$	$35.6\text{G}\times3$	26.6M	51.0	79.3
META			$8\times3\times10$	$35.6\text{G}\times30$	26.6M	**52.1**	80.2

[1] Not counting Latent Motion Code (LMC) module's parameters.
[2] Re-counted using official public code for digit precision.
[3] Our implementation using official public code.

When comparing our work with recent Transformer-based state-of-the-art methods, however, META is inferior to those methods presented in the middle part. Without considering FLOPs and the number of parameters, META is 3.3% less accurate than UniFormer-B in terms of top-1 accuracy and 1.3% lower than EAN RGB+LMC. According to [64], Transformer's strength comes from its architecture, which was built to aggregate global information earlier due to self-attention. Besides striking differences in architecture concept, we found that UniFormer used Kinetics as its pre-trained model while we only pre-trained our model from ImageNet. Moreover, our model takes eight frames as the input image, while those Transformer-based models require more frames than us to serve as an input image.

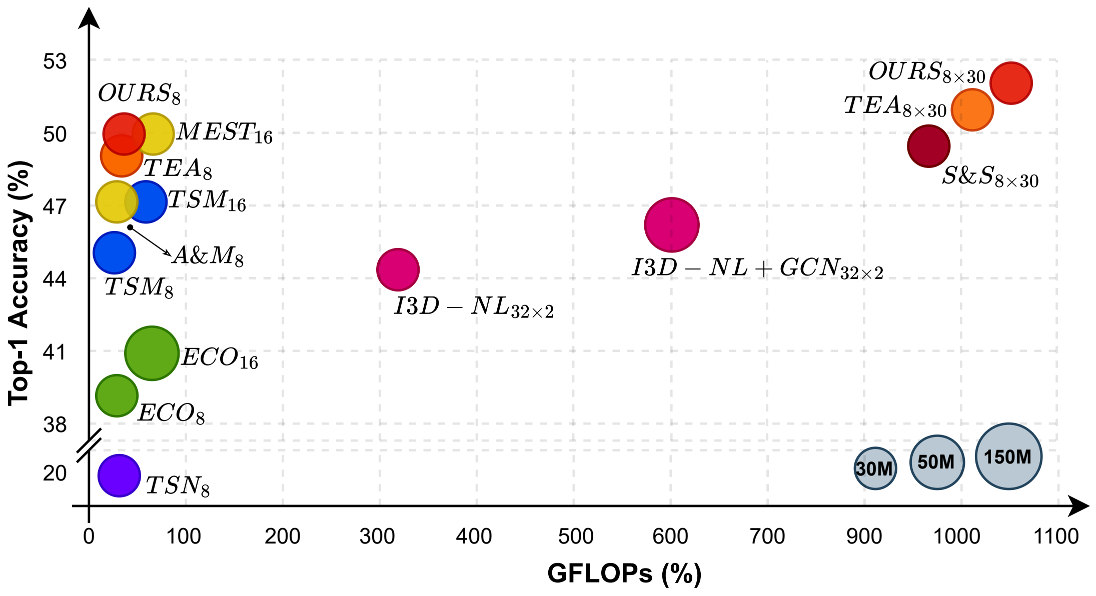


Fig. 4.13 Ball chart reporting the top-1 accuracy *vs.* computational complexity (in GFLOPs). The size of each ball indicates model complexity. "A & M" corresponds to ACTION-NET [88] and MEST [98], while "S & S" denotes STM [41] and SMNet [94], respectively. We merge their icon since they share similar number of accuracy and GFLOPs.

Figure 4.13 shows a visual comparison of CNN-based techniques using a ball chart. We report the top-1 accuracy with respect to floating-point operations in gigabyte (GFLOPs). Accuracies are calculated using only center crop and single forward pass unless otherwise specified. The plot demonstrates how we consistently excel over comparable works while keeping FLOPs to a minimum level (only 1.08$\times$ as many as TSM). For our method and TEA, we find that total accuracy may be improved by a factor of $\pm$1.05, at the expense of computational costs that increase to well over a thousand GFLOPs. The plot shows that overall, 2D CNNs may outperform 3D CNNs when the 2D-based network is provided with sufficient temporal feature learning.

Table 4.4 The comparison result of META against other state-of-the-art methods on the Jester dataset. The listed methods used ResNet-50 as the backbone, except for TRN Multiscale used BNInception. Scoring was done using a validation set.

Methods	**Frame**	**FLOPs**	**Top-1 %**	**Top-5 %**
TSM [53]	8	33.0G	97.0	**99.9**
MFNet-C50 [48]	7	-	96.1	99.7
TRN Multiscale [100]	8	16.0G	95.3	-
TSN [53]	8	33.0G	83.9[1]	99.6[1]
STM [41]	8	33.3G	96.6	**99.9**
TEA [52]	8	35.1G	96.5[2]	99.8[2]
ACTION-NET [88]	8	34.7G	97.1	99.8
ACTION-NET [88]	16	34.7G	97.1	**99.9**
META (Ours)	8	35.6G	**97.1**	99.8

Jester. Likewise, Jester is classified as a temporal-related dataset. Our experiment result is provided in Table 4.4. Clearly, our work demonstrates superiority on the Jester dataset in terms of Top-1 accuracy compared to the baseline (97.1% vs. 97.0%) and other state-of-the-art methods except for ACTION-NET, where we obtained the same accuracy. Our interesting finding according to this table: both META and ACTION-NET, which are equipped with a motion representation module, achieved only a slightly higher accuracy than TSM ($\triangle$0.1% of accuracy) with no motion representation module in it. Though admittedly, we need further experiments to verify this, we may think that motion representation has less meaning for this dataset. Also, results from TEA and STM confirm our thought as both methods proposed this module and the performance is inferior compared to ours ({96.5%, 96.6%} vs. 97.1%).

Moments in Time Mini. Unlike the above datasets, this dataset possesses characteristics of temporal-related and scene-related datasets. The performance of our proposed work is still impressive and Table 4.5 confirms this. We achieved the highest accuracy in terms of top-5 accuracy. While we are inferior in terms of top-1 accuracy compared to IR-Kinetics400, we want to emphasize that IR-Kinetics400 utilized a Kinetics-400[11] dataset as their pre-trained weights, whereas we only used ImageNet pre-trained weights. Moreover, their base model combines Inception and ResNet, while we only adopt a single base model, i.e., 2D ResNet-50.

Table 4.5 The comparison result of our work against other CNNs on the Moments in Time Mini validation set.

Methods	**Backbone**	**Top-1 %**	**Top-5 %**
TRN from [10]	BNInception + InceptionV3	26.1	48.5
P3D from [10]	P3D ResNet	14.7	33.4
P3D-Kinetics from [24]	P3D ResNet	26.3	-
IR-Kinetics from [24]	Inception + ResNetV2	**30.3**	-
I3D-DenseLSTM [42]	I3D + ResNext	26.5	52.4
META (Ours)	2D ResNet-50	27.4	**53.2**

4.4.6 Discussion

Examples of Successful Prediction

We illustrate some accurate predictions of META over other works in Figure 4.14. To obtain the probability score in (a), we re-train the model using the official code publicly available[2], whereas we only load the model with the official weights[3] for (b).

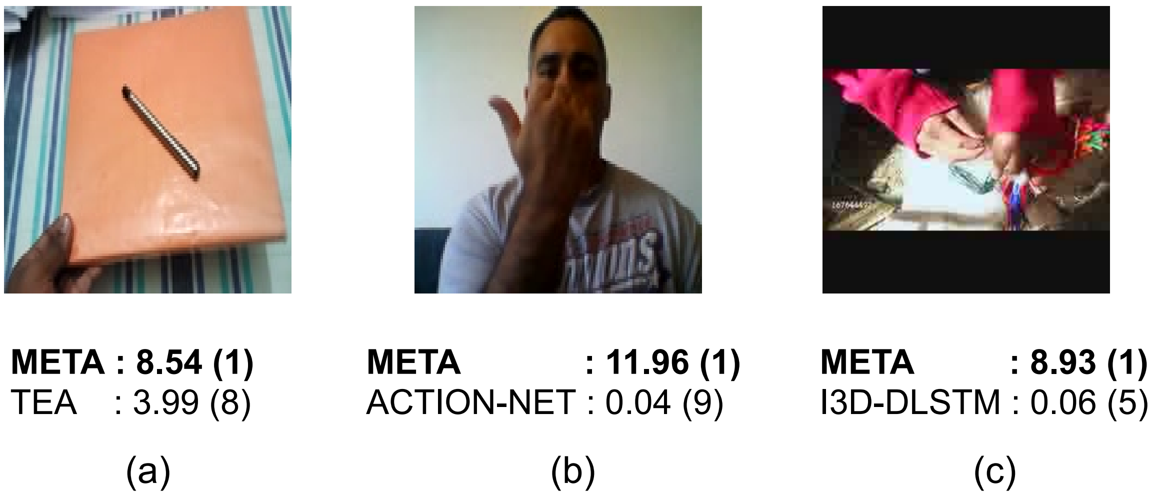


Fig. 4.14 Examples of prediction results showing three frames and its probability scores (left-right): Something-something v1 ("*Lifting something with something on it*"), Jester ("*Pulling hand in*"), and Moments in Time Mini ("*Tying*") datasets.

[2]https://github.com/Phoenix1327/tea-action-recognition
[3]https://github.com/V-Sense/ACTION-Net

In all scenarios, we confidently achieve top-1 accuracy (indicated by a number in parenthesis) with substantial difference in probability score, whereas other works rank below ours. For instance, (a) informs that META exactly predict an action of "*Lifting something with something on it*" whilst TEA measures such action in 8th out of softmax outputs in descending order. This fact demonstrates our predominance over existing related works in three datasets.

GradCAM Analysis

In addition, we visualize the significant region extracted by TSM, TEA, and our META for the action "*Lifting something with something on it*" in Figure 4.15. Features extracted by each method are visualized by using GradCAM. Compared to TSM and TEA, we can see that our method has a smaller interest region, meaning our method can focus on an object. Whether it is not apparent that a smaller interest region is a key here, our method achieves top-1 accuracy compared to TEA. An interesting fact in the visualization is that the TSM can also reason the hand, while other methods fail to detect a hand. It can be seen in the left corner in the third and fifth columns.

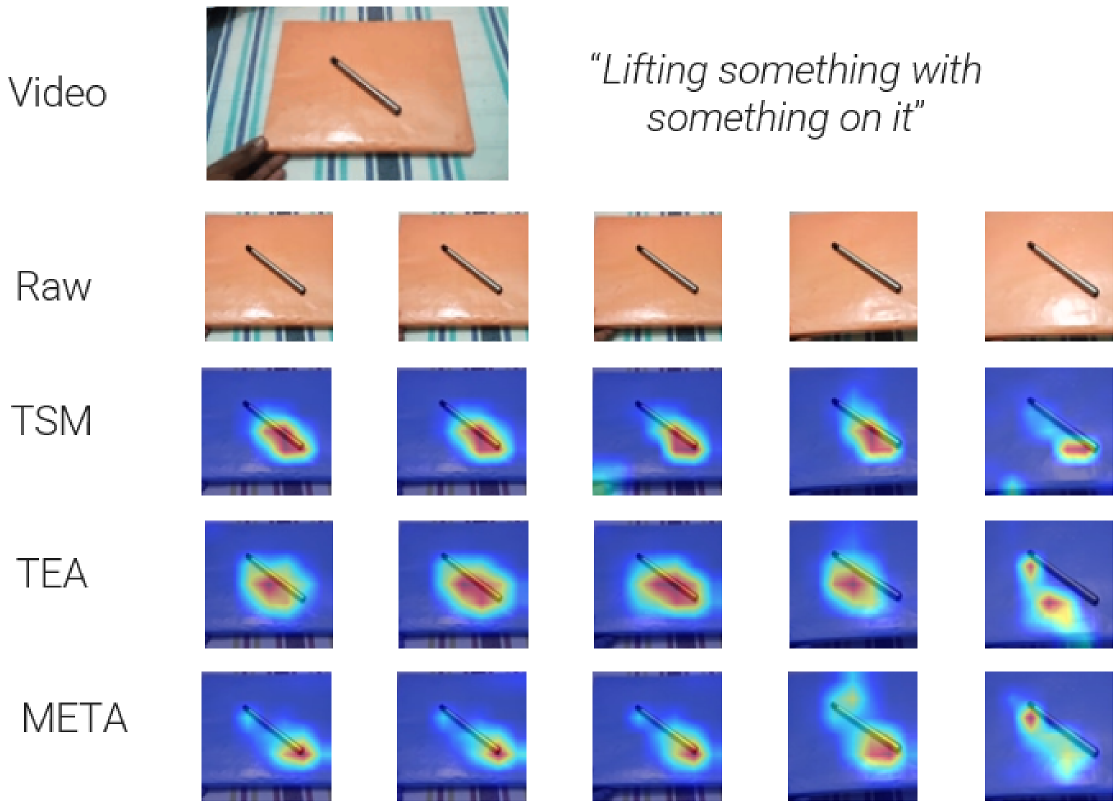
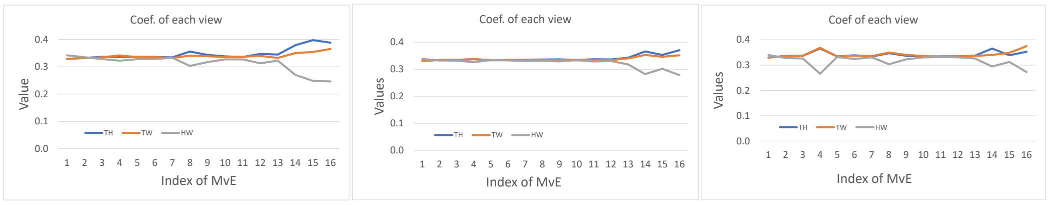


Fig. 4.15 Visualization of GradCAM using several methods.

Coefficients (α) Analysis in Multi-view Submodule

We also investigate the magnitude of the learned coefficients and quantify the contribution of different views. Specifically, for each MvE layer, the mean coefficient of each view is computed on the validation set. Figure 4.16 shows the comparison of the importance of different views in all three large-scale datasets.



(a) (b) (c)

Fig. 4.16 Subfigures a, b, and c illustrate values of MvE coefficients for the Something-something v1, Jester, and Moment in Time Mini datasets, respectively.

The coefficient of the HW view measures the importance of appearance feature, while those of the TH and TW views measure the importance of temporal motion cues. The overall importance of each view can be measured by averaging the mean coefficients of all MvE layers. On the Something-something v1 dataset, the mean coefficients of the TH, TW, and HW views are 0.35, 0.34, and 0.31, respectively. While on the Jester dataset, they are 0.34, 0.34, and 0.32. Lastly, on the MIT mini dataset, the TH, TW, and HW values are 0.34, 0.34, and 0.32, respectively. Hence, spatial and temporal feature plays a significant role in three datasets. Nevertheless, we notice that on the Something-something v1 dataset, the spatial feature has less meaning than the temporal feature, with about 4% lower. Whereas, in the other two datasets, the spatial feature has 2% less than the temporal feature. In summary, we observe that the bottom layers of the network focus on spatial and temporal feature learning equally, while the top layers attend more to temporal feature aggregation.

4.5 Summary

This chapter presents a novel building block to overcome the existing problems for the video action recognition task by designing three submodules to construct a META block and integrating it into each residual block of 2D ResNet-50. The proposed block includes excitation of motion and multi-view features followed by densely connected temporal aggregation. While retaining modest computations, our META achieves competitive results on three large-scale datasets compared to its 2D/3D CNN counterparts.

Compared with recent Transformer-based networks, our work still achieves competitive results on the Jester dataset while being inferior on the Something-something v1 dataset.

In the future, we would like to investigate another fusion approach, i.e., channel concatenation in the DCTA submodule, so that all layers are connected, and the current input is the concatenation of the preceding layers. This fusion will guarantee that new information is added to the collective knowledge.

Chapter 5

Deep Fusion Schemes of Temporal Gaussian Mixture

5.1 Introduction

Unquestionably, action recognition is currently a topic of active research due to the challenges researchers must overcome in this field and the importance of action recognition applications in our daily lives. Such applications include criminal detection, more timely alert systems for natural disasters, video summarization, and video recommendation systems. One of the challenges in classifying an action in a video is accurately capturing the hidden temporal relation between frames in addition to capturing the spatial semantics. To take both aspects into account, the filters of the convolutional layer must operate on 3-dimensional data (3D), in contrast with the more general approach for image classification that uses only 2-dimensional (2D) convolution.

3D convolution is effective in capturing both temporal and spatial dimensions. Although a video consists of spatial and temporal aspects, there have been several attempts to understand the variations in actions through the use of 2D convolution without sacrificing accuracy [41, 30]. Such 2D convolution helps the computer to train faster and reduce the inference times than would be the case using 3D convolution.

A different approach has been proposed to use optical flow fields as an additional input modality while keeping the RGB images as the primary input. This modality focuses only on pixel displacement over time, removing unnecessary spatial information in frames. Optical flow is computed by taking the difference between two consecutive frames, providing helpful information regarding the motion.

Because more than one modality is used, several studies have used at least two different architectures to accommodate different modalities to learn spatial and temporal information separately. Optical flow components from the RGB images are extracted and fed into different architectures, referred to as the "branches" for each separation in this paper. Each branch is expected to be complementary to the other branches. Previous works based on this approach [18, 12, 96] have obtained impressive results. Different architectures that use the recurrent model to understand data sequences have been introduced. Considering frames in a video as a sequence of items of information, the recurrent model aims to learn the temporal relations contained in frames.

5.2 Related Work

5.2.1 Three-Stream CNNs

Deep convolutional neural networks (ConvNets) have demonstrated an impressive capacity for learning and representing the visual feature. With the advent of ConvNets, significant progress has been achieved in video-based action recognition. However, mainstream ConvNets, including two-stream ConvNets and 3D ConvNets, utilized for video-based action recognition, still cannot encode fine-grained information.

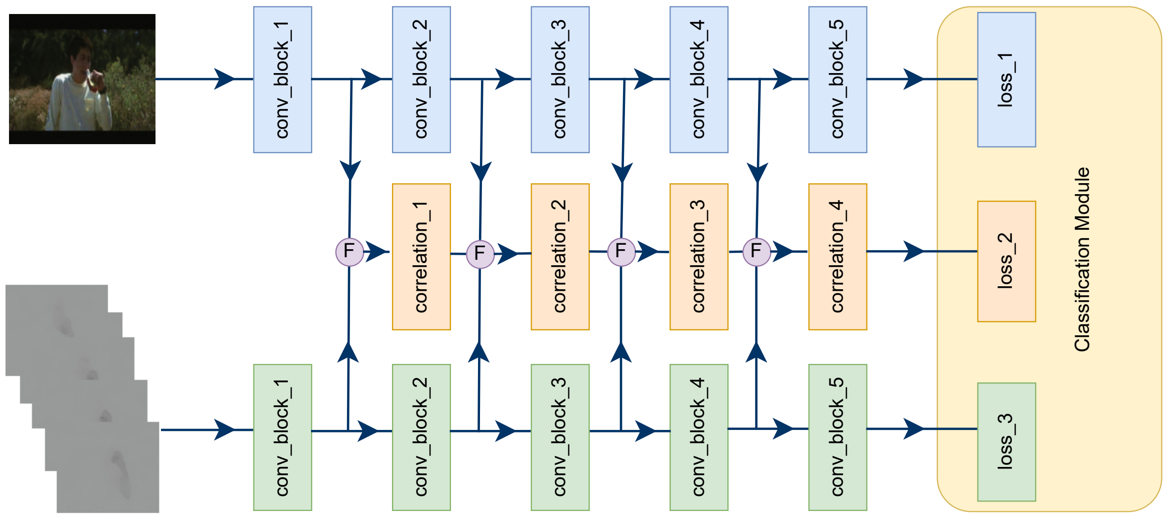


Fig. 5.1 The three-stream network's overview at multiple levels (MLTSN). Each of the MLTSN's three streams—spatial, temporal, and multi-level correlation streams (MLCS)—can produce high-level features for classification.

Based on this condition, research led by Lv *et al.* [54] proposed a Multi-level Three-Stream based on CNNs to yield refined features. As illustrated in Figure 5.1, a newly created branch termed Multi-level Correlation Stream consists of four correlation blocks that process fine-grained features from spatial and temporal streams.

Another approach by Khalid *et al.* [44] adopted the same three-stream architecture of the deep neural network. Rather than merging two modalities, their approach utilizes the pose modality as an input to the pose stream. They suggested a novel video-based action detection system that uses complementing cues to solve the challenging issue of high variation and complexity of data. Additional pose features are researched and used with the successful two-stream networks for action classification to improve comprehension of human action more abstractly and semantically. With our suggested pre-processing module, practices can include noisy estimated poses and ground truth poses in the framework. Figure 5.2 illustrates their approach. Two stream networks are proposed to extract visual and motion features simultaneously, significantly improving classification accuracy. The pose stream takes 3D tensors of encoded human joint poses to train a CNN-based network. Finally, complementary cues for action recognition, i.e., appearance, optical flow, and posture features, are analyzed and fused to handle varied action classes.

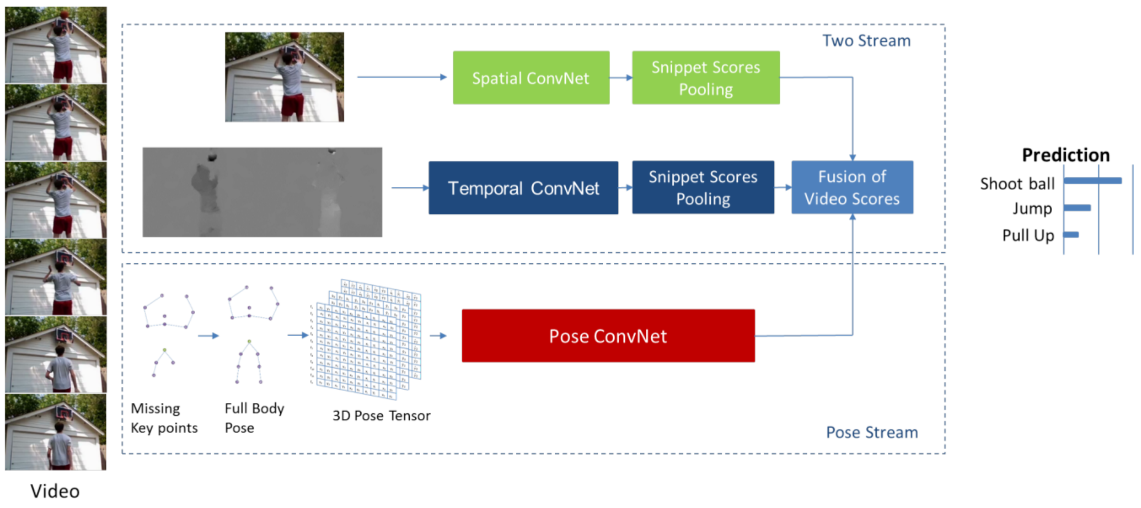


Fig. 5.2 Three streams convolutional neural network (TSCNN) is formulated by fusing scores from Pose ConvNET with video-based scores from spatial and temporal streams. Image is taken from [44].

5.2.2 Feature Fusion Strategies

Feichtenhofer *et al.* [17] introduces the concept of Slow and Fast pathways, as shown in Figure 5.3. One pathway operates at low frame rates and slow refreshing speeds and is intended to capture semantic information that images or a few sparse frames can provide. The other pathway, which operates at a high temporal resolution and fast refreshing speed, captures motion that changes quickly.

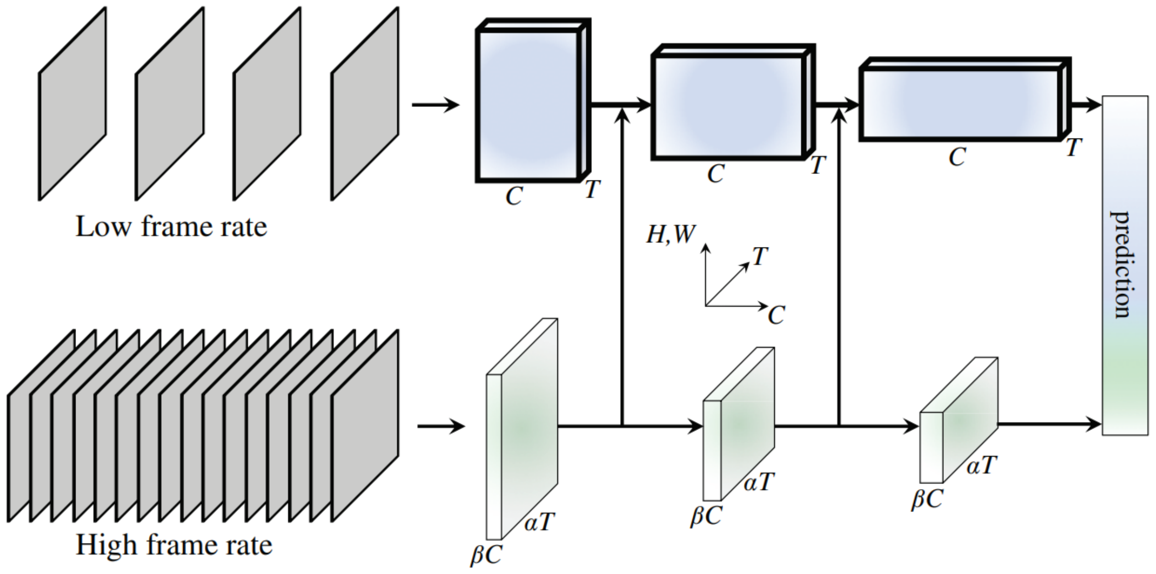


Fig. 5.3 A SlowFast network consists of two pathways: a Slow pathway has a low frame rate and low temporal resolution whereas a Fast pathway has $\alpha\times$ higher temporal rate with $\beta\times$ lower channel capacity. Image is taken from [17].

This method is designed to be very lightweight despite having a high temporal rate. This design is motivated by the fact that the first pathway may offer spatial information less redundantly. In contrast, the second pathway is meant to have fewer channels and a lower ability to comprehend it. Because of their different temporal rates, they refer to the first as a Slow pathway and the second as a Fast pathway.

To realize correlations between two pathways, the author constructs several lateral connections. By creating lateral connections between the two pathways, one pathway can be aware of the representation that the other pathway has learned. Lateral connections are the main method for combining various levels of spatial resolution and semantics in image object recognition. For each "step," they attach a single lateral link between the two channels (see Figure 5.3). These connections, specifically for ResNets [31], come right after pool1, res2, res3, and res4. The lateral connections modify the tensor shape to meet the two paths' different temporal dimensions. They experimented with one-way links that combine elements of the Fast and Slow pathways. They have also tried out bidirectional fusion and discovered comparable outcomes. On the output of each pathway, a global average pooling is then carried out. The input to the fully-connected classifier layer is then created by concatenating two pooled feature vectors.

5.2.3 Temporal Gaussian Mixture

In video action recognition tasks, it is essential to learn temporal relationships for better classifying an action. A work by Piergiovanni *et al.* [60] introduced a novel method to learn long-range temporal evolution. They described how to effectively capture longer-term temporal information in continuous activity videos using a novel convolutional layer called the Temporal Gaussian Mixture (TGM) layer. The TGM layer is a temporal convolutional layer controlled by a considerably smaller set of fully differentiable parameters, such as the placement and variance of Gaussians.

Figure 5.4 illustrates the overall process inside the TGM layer in a single branch. The kernel is a constrained kernel governed by the variance of the Gaussian, i.e., a center μ and a width σ for which the values are in the positive range.

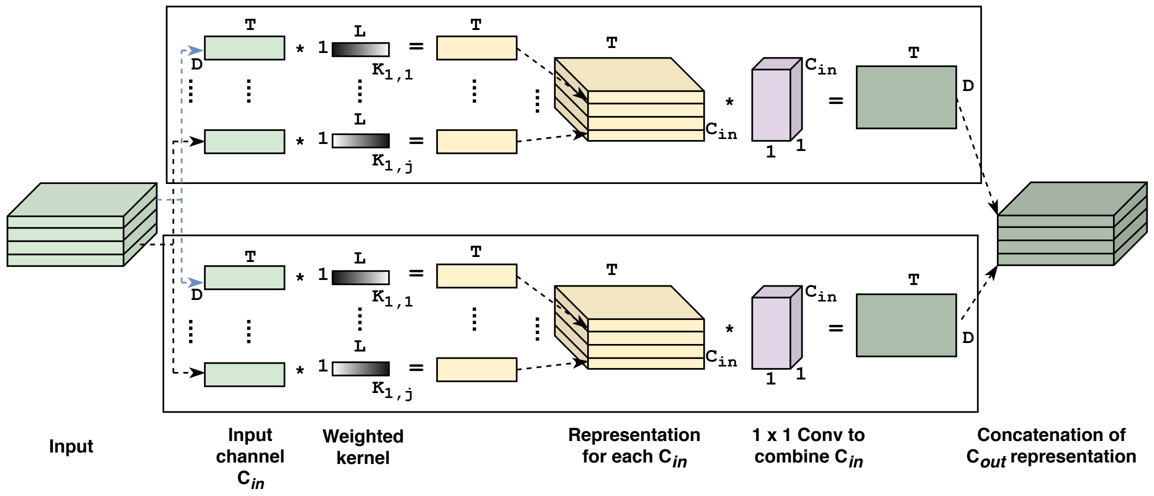

T
D
L
1
K1,1
K1,j
=
*
Cin
1
Input
Input channel Cin
Weighted kernel
Representation for each Cin
1 x 1 Conv to combine Cin
Concatenation of Cout representation

Fig. 5.4 Inside the TGM layer. A Gaussian weighted kernel with length L is multiplied by each input channel C_{in} to produce a tensor with the shape of $C_{in} \times D \times T$. The figure illustrated above is taken from [60].

Figure 5.5 shows that this layer also includes an attention mechanism widely used in computer vision and language processing. Soft attention is applied to each Gaussian distribution to enable the layer to focus on the relevant parts in a temporal sequence.

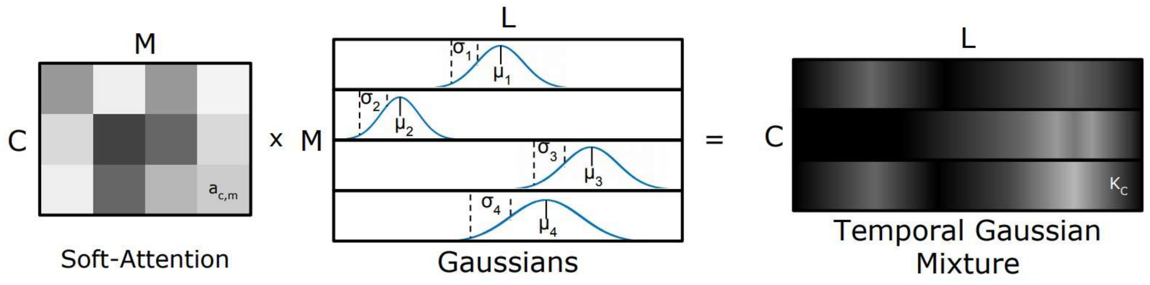


Fig. 5.5 A TGM kernel with multiple temporal convolutions C and length L is computed from learnable attentive parameters and multiple Gaussian distributions. M denotes the number of Gaussian distributions. There are two constrained variables for each Gaussian distribution: a center μ and a width σ. The image is taken from [60].

5.3 Proposed Deep Fusion of Temporal Gaussian Mixture for Multi-Label Multi-Class Problem

We introduce our new architecture that consists of several Gaussian kernel-based branches. Our proposed model use two modalities, RGB data and optical flow. Thus, we have two branches based on their input types, namely, spatial and temporal, to classify the actions. Given the outputs from the base CNN, represented as $F \in \mathbb{R}^{T\times1\times1\times D}$, we first propagate F to the spatial and temporal branches. T and D are the temporal length and the feature maps' dimensions, respectively. We set the number of the TGM layers to three for each branch, identical to that of the referenced work [60]. Inside the TGM layer, we learn a set of Gaussian mixture kernels. For each input channel $j \in [1,\, C_{in}]$ and each output channel $i \in [1,\, C_{out}]$, the associated filters will convolve on the temporal input feature x and map the resulted channels into a single channel to produce s_i:

$$s_i = (x * K_{i,j}) * w_i \tag{5.1}$$

where K $= [K_1,\, K_2, \ldots,\, K_{c_{out}}]$ denotes the Gaussian mixture kernels corresponding to each input and output channel, and w_i is a 2D convolution with 1×1 kernel size and one as the output channel, followed by the rectified linear unit (ReLU) activation function. Notation for channel-wise operation and ReLU activation function is ignored for simplicity. The obtained s_i is then stacked on the channel axis to produce $S_{rgb} = [s_1, s_2 \cdots, s_{C_{out}}]$ with a dimensionality of $C_{out} \times D \times T$, where C_{out} is the number of the output channels. C_{in} and C_{out} can be considered as hyperparameters. We set C_{out} to four in this work. The value of D is consistent throughout all TGM layers (i.e., 1,024).

Unlike the original work, we argue that the maximum function accentuates temporal features' distinctive, important aspect. Thus, for the temporal branch (S_{flow}), we replace 1×1 convolution + ReLU (Equation (5.1)) with the aggregate function. Mathematically, this can be formulated as follows:

$$s_i = \max(x * K_{i,j}) \tag{5.2}$$

In Equation (5.2), we replace w with the max function operating on the channel axis. The output s_i is then appended along the channel axis to obtain S_{flow}, which is the $C_{out} \times D \times T$ representation, identical to that of S_{rgb}. Then, S_{rgb} and S_{flow} are fed forward to the next layer. In addition, the output of the base model CNN and the last TGM layer are concatenated (see illustration in Figure 5.6).

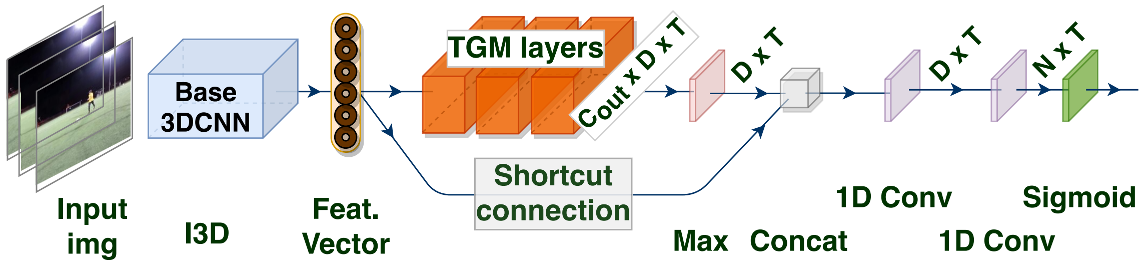


Fig. 5.6 The output from 3DCNN is concatenated with the output from the last TGM layer. This shortcut connection prevents information loss during the learning phase. Squeeze and permutation are necessary for the output of base 3DCNN ($T \times 1 \times 1 \times D \rightarrow D \times T$) prior to the concatenation. This figure illustrates only one branch, and thus no fusion occurred.

We want to highlight the implementation of the TGM layer in our proposed spatio-temporal branch. In the original paper, the authors implement the TGM layer only on the RGB and optical flow modality. In contrast, the TGM layer is utilized on mixed modalities for fine-grained features in our implementation.

This section presents three different fusion methods utilizing the TGM layer. We will list each of them and discuss their characteristics.

5.3.1 Spatial and Temporal Fusion

As observed from Figure 5.7, a fusion is introduced between the TGM layers. This type of combination is similar to those in work by Feichtenhofer *et al.* [17]. The two lateral connections are established from the spatial branch to the temporal branch to merge a meaningful representation of the RGB data. In contrast to their work, where the outputs are transformed before fusing, we merely add the RGB and optical flow features because the shape and length of those two features already match. Given S^{rgb} and S^{flow} that are the outputs of the previous TGM layers in each branch, a new S^{flow} is determined according to the following equation:

$$S_{i+1}^{flow} = S_i^{flow} \oplus S_i^{rgb},\ i = [0, 1] \tag{5.3}$$

where i is the index where fusion occurs.

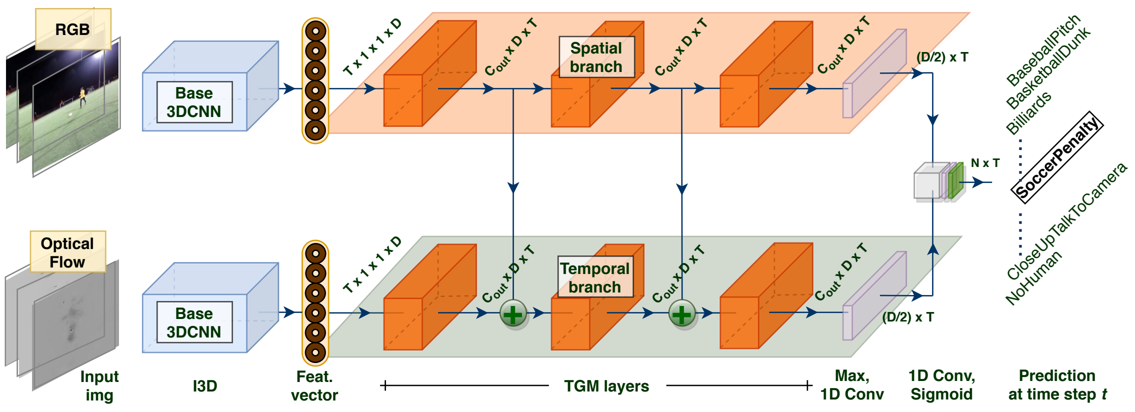


Fig. 5.7 Detailed architecture of fusion of spatial and temporal streams.

5.3.2 Early Spatio-Temporal Fusion

In this approach, we construct a new pathway from the existing branches. We introduce a spatio-temporal branch with a fusion occurring at the beginning of the branch (see Figure 5.8). We want our model to learn both spatial and temporal information separately and spatio-temporal information simultaneously. We are confident that the model will benefit from this fusion. Formally, a new spatio-temporal branch (S^{st}) results from the element-wise addition of S^{rgb} and S^{flow}.

$$S^{st} = S_i^{rgb} \oplus S_i^{flow}, i = 0 \tag{5.4}$$

5.3.3 Multi-Level Spatio-Temporal Fusion

In contrast to Proposed #2, we carry out a fusion strategy at several levels, as illustrated in Figure 5.9. We argue that each level of the TGM layers produces different temporal activity patterns. As described in Equation (5.5), given features S_{rgb} and S_{flow}, we

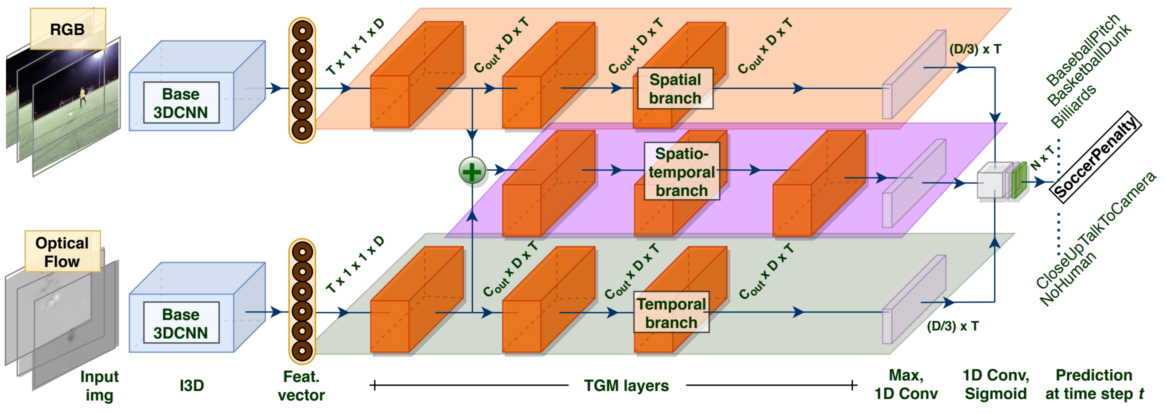


Fig. 5.8 Detailed architecture of fusion of spatial and temporal streams.

perform element-wise addition at different levels to form the spatio-temporal branch, denoted as S_{st}.

$$S_i^{st} = S_i^{rgb} \oplus S_i^{flow}, i = [0, 1, 2] \tag{5.5}$$

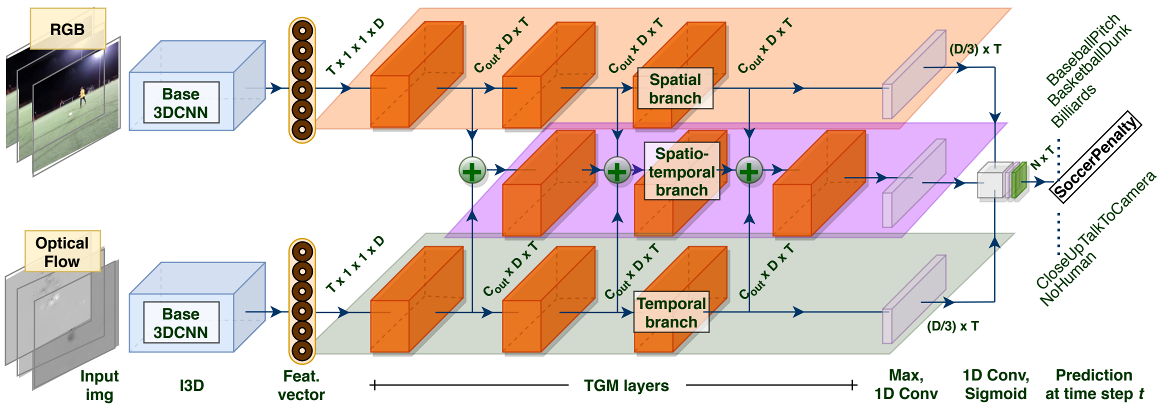


Fig. 5.9 Detailed architecture of fusion of spatial and temporal streams.

5.4 Experiments and Evaluation

In this section, we describe our experiment in detail. We demonstrate the implementation of the TGM layer in each branch and simultaneously fuse the spatial and temporal branches to achieve a positive result. Moreover, we conduct some ablation studies to emphasize the benefit of the *Max* function over 1×1 convolution inside the TGM layer and to investigate the benefits of the weighted branch scheme.

5.4.1 Dataset

We conducted some experiments on the untrimmed, multi-label MultiTHUMOS dataset [95] to investigate the effectiveness of the spatio-temporal branch. This dataset is an extension of the well-known THUMOS dataset, with its number of action classes extended from 20 classes to 65 classes. Verbs are taken from the original THUMOS with several additions of various activities.

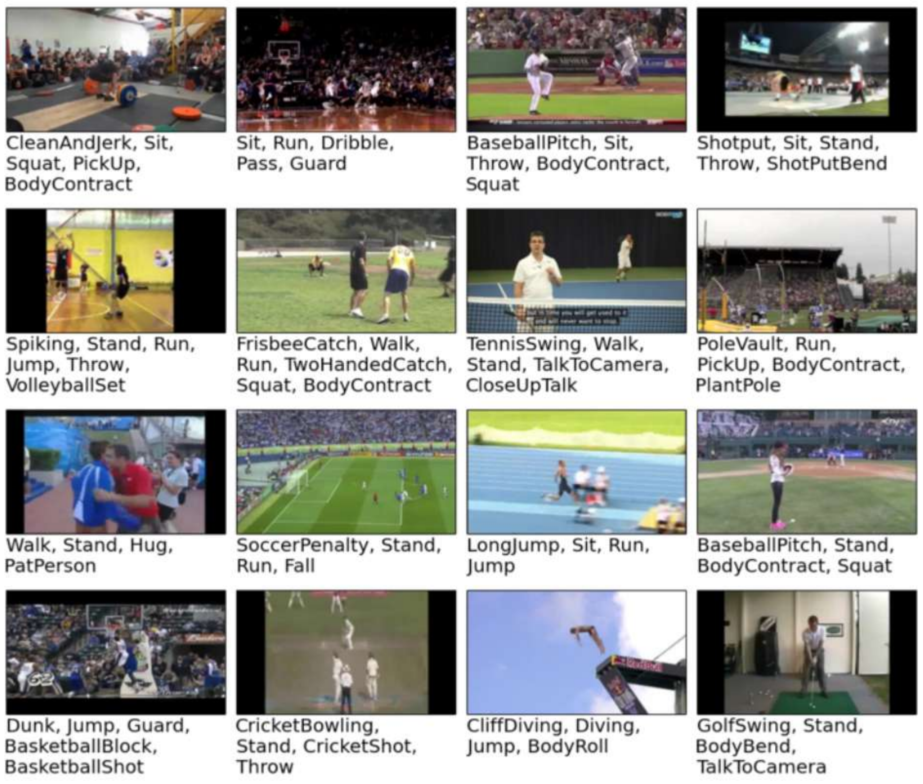

CleanAndJerk, Sit, Squat, PickUp, BodyContract
Sit, Run, Dribble, Pass, Guard
BaseballPitch, Sit, Throw, BodyContract, Squat
Shotput, Sit, Stand, Throw, ShotPutBend
Spiking, Stand, Run, Jump, Throw, VolleyballSet
FrisbeeCatch, Walk, Run, TwoHandedCatch, Squat, BodyContract
TennisSwing, Walk, Stand, TalkToCamera, CloseUpTalk
PoleVault, Run, PickUp, BodyContract, PlantPole
Walk, Stand, Hug, PatPerson
SoccerPenalty, Stand, Run, Fall
LongJump, Sit, Run, Jump
BaseballPitch, Stand, BodyContract, Squat
Dunk, Jump, Guard, BasketballBlock, BasketballShot
CricketBowling, Stand, CricketShot, Throw
CliffDiving, Diving, Jump, BodyRoll
GolfSwing, Stand, BodyBend, TalkToCamera

Fig. 5.10 Some videos of the MultiTHUMOS dataset. Each video may contain multiple activities; hence, the task is categorized as a multi-label multi-class classification problem. The image is taken from [95].

In total, we examined 30 hours of duration from 400 videos. This dataset contains 38,690 annotations, with every frame having 1.5 labels on average, notably higher than THUMOS (0.3 per frame). Each video has 10.5 action classes, increasing sharply from the 1.1 of THUMOS. Examples of the videos are illustrated in Figure 5.10.

5.4.2 Experimental Setup

Before video feature extraction, we first extract all of its RGB frames and optical flow. We crop all the extracted frames to a window size of 224 $\times$ 224 at the center to match the base model's input dimensions. We also normalize all pixels to have values with a range of $[-1 \ldots 1]$. In addition, we use the well-known TVL1 algorithm [97] to compute the optical flow.

The next step is to extract the video features from the image data. Before extracting, we load our model with the weights pre-trained on the ImageNet and the Kinetics dataset. This strategy is a common transfer learning technique. We propagate onward our collection of frames via the I3D network to obtain the activations. We select the last average pooling *AvgPool3d* of I3D to serve as an endpoint logit for feature extraction, the same as the referenced work. The extracted shape of each video is $T \times 1 \times 1 \times D$. The value of T can be varied depending on the length of a video, whereas D is consistent for all videos (i.e., 1,024). We note that the TGM layer can accept arbitrary temporal lengths.

The output features are then formatted as NumPy arrays and saved to a disk. If the number of frames is excessively high (i.e., a video has a long duration), then we divide it by the threshold (in our case, we set it to 100) and forward propagate the current chunk to the I3D network. The value of 1,024 represents the number of channels. The term "channel" here can be ambiguous. We typically use this word to describe the number of feature maps, whereas "channel" in the TGM layer, represented by C_{in} and C_{out}, refers to the number of Gaussian mixtures.

5.4.3 Evaluation Measure

To measure the performance, we define the mean average precision (mAP) as follows:

$$mAP = \frac{1}{N} \sum_{c=1}^{N} AP_c \tag{5.6}$$

According to Equation (5.6), the mAP is calculated by summing each AP (AP_c) and then dividing by N (the number of queries), whereas the AP for a specific class is computed using *precision* at each relevant position n:

$$AP = \frac{\sum_{n=1}^{M} \text{Precision}(n)}{M} \tag{5.7}$$

where M is the total number of actions predicted.

5.4.4 Result & Analysis

Clearly stated, we aim to improve previous work [60] by implementing a variety of fusion schemes to increase the mAP. We postulate that a fusing mechanism is essential for accuracy.

This section discusses how well the proposed models classify the frames into predefined action classes. To accomplish this, we plot the prediction into the temporal region with the X-axis being the time axis (see Figure 5.11). This figure enables us to determine the accuracy of each proposed model by comparing the nonblack regions (the proposed models) with the black region (the ground truth) in the video of the volleyball game. It is observed that our proposed models exhibit good performance in some activities. We plot several video frames to describe the activities of "*Run*" and "*VolleyballSpiking*".

In the *Run* activity, surprisingly, our proposed models significantly improve the performance while the baseline model fails to predict the activity. Figure 5.11 shows that a person with a red dot is performing *Run* before jumping and hitting the ball. In the right-hand layout, our proposed models correctly predict this activity; the red temporal region stretches for some time, whereas the baseline model misses the prediction.

We also show the advantage of fusing the TGM layers in *VolleyballSpiking*. While the baseline model fails to accomplish the prediction, our proposed models enjoy the benefits of the fusing mechanism that can locate and classify the spiking activity in video frames. Again, a man with a red dot indicator is carrying out a volleyball spike. At the same time, this man is also performing the *Jump* activity. Our models predict *Jump* activity with a higher confidence level than the base model: the temporal regions are drawn continuously for our proposed models.

Despite successful predictions, we also observed the occurrence of misprediction. As shown for the *Sit* activity, none of the models can classify this activity even when it is present in reality. We suspect that our proposed models have difficulty learning its temporal structure because no changes are occurring temporally (i.e., the object/person is motionless). Furthermore, it is possible that the object is relatively tiny for detection purposes, and thus that our proposed models fail to notice it. We note that a false-positive prediction also occurred, as was found for the *VolleyballSpiking* and *VolleyballBlock* activities. In earlier times of these activities, all models predict a non-existence action, in contrast with the prediction after some t time.

We also observe that all of the proposed models are accurate in *Fall*, *VolleyballSet*, and *NoHuman* class. Additionally, we are confident that the fusion of two modalities, regardless of how they are fused, exhibits outstanding performance compared to the baseline. This assumption suggests that exploring other fusion techniques (e.g., temporal fusion, as mentioned earlier) will be beneficial.

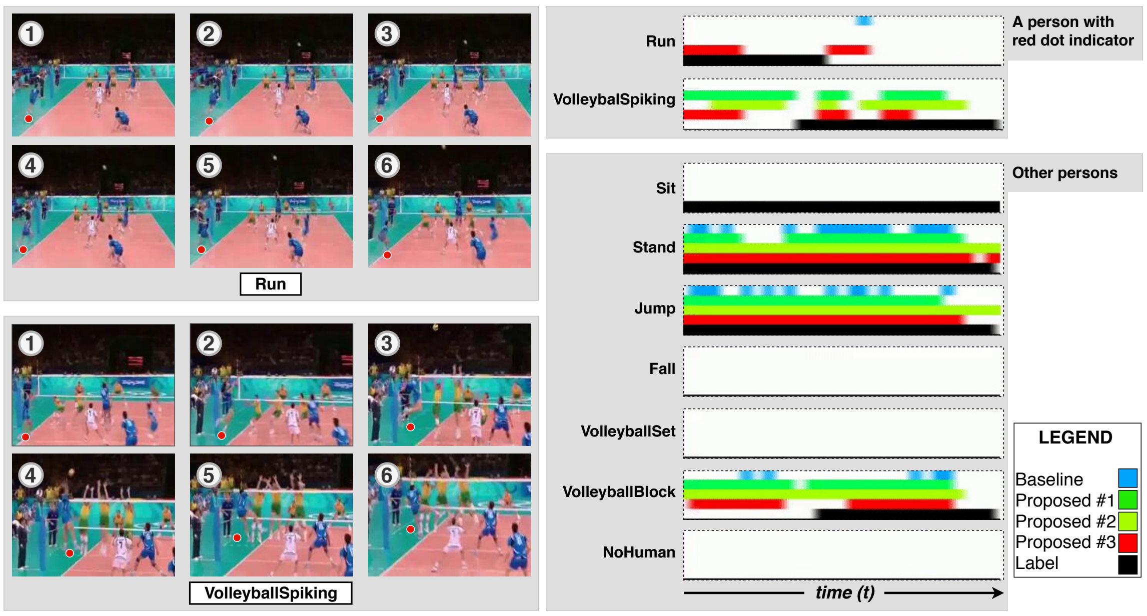


Fig. 5.11 An example of various activities in a video of a volleyball match. There are five temporal regions to illustrate the lifespan of an activity. Temporal regions in black and blue show the ground truth (label) and the baseline model, respectively. The other colors show the proposed models. We show video images that describe the two actions, "*Run*" and "*VolleyballSpiking*". A red dot indicator is placed below a person performing a specific activity at time t. It is observed that interpolating semantic information between two branches improves performance. Best viewed in color.

5.4.5 Comparison with Related Work

This section compares our three proposed models with the baseline and state-of-the-art methods. We evaluate three types of fusions and conduct short analyses based on the results.

An examination of the results presented in Table 5.1 shows that by integrating the information of the spatial branch into the temporal branch, the improvement of Proposed #1 over the baseline is marginal (0.9% higher). This fact confirms the advantage of the one-way information sharing of two branches. In the following approach, the model Proposed #2 does benefit from having a newly created branch in the form of a 1.5% improvement over the baseline. This value certifies that inserting a new branch and simultaneously optimizing the weights of all branches will improve the

Table 5.1 Comparison with other state-of-the-art methods on a popular benchmark MultiTHUMOS dataset. Even with fewer output channels (C_{out}) for each TGM layer, we consistently outperform the baseline and other methods.

Method	**mAP (%)**
Two-stream by Yeung *et al.*[95]	27.6
Two-stream + LSTM by Yeung *et al.*[95]	28.1
Multi-LSTM by Yeung *et al.*[95]	29.7
Predictive-corrective by Dave *et al.*[13]	29.7
SSN by Zao *et al.*[99]	30.3
Ours (Proposed #3)	**37.1**

performance to some extent. Our last and best model, Proposed #3, outperforms the baseline method by 2.9%. We believe that this model captures more complex, nonlinear temporal features and thus contributes significantly to the model's performance.

We can conclude that a fusion mechanism demonstrates consistent performance over the baseline model and other existing models on the MultiTHUMOS dataset.

5.4.6 Discussion

This section describes experimental studies on MultiTHUMOS. We conducted several comparisons to identify promising strategies. We experiment with various channel combination strategies. We also conduct ablation experiments of weighted schemes for spatial and temporal branches. In addition, we discuss a possibility to incorporate the previous approach (META) for a multi-class action classification problem into a multi-label action classification task.

Custom Channel Unification

Referring to Figure 5.4, we observe that the original version applies a 2D convolution to combine temporal reasoning features on the channel axis. This 1×1 2D convolution is designed to combine all of the C_{in} into one channel.

To the best of our knowledge, there are more straightforward, less expensive, yet effective techniques for achieving channel combination: summation, average, and maximum. These functions have no parameters to optimize and thus have a lower computational cost. We performed several experiments to determine the effectiveness of each function compared to those obtained in the original work. The results are presented in Table 5.2. We observed that 1×1 convolution is more beneficial to the spatial branch and, conversely, is not helpful for the temporal branch. Surprisingly, the TGM layer with the maximum function performs better on the temporal branch. We hypothesize that this function shows slight improvement because each pixel in the temporal branch corresponds to small changes; thus, the maximum function ensures that the model takes only distinctive pieces of representation in each channel. Consequently, the subsequent layer processes more fine-grained temporal features.

Table 5.2 Results for different channel combinations. Interestingly, different modalities yield different results for the same operation. This table also emphasizes that the RGB images are a more critical modality than the optical flow in terms of accuracy. Note that "3" in 3TGM refers to the number of TGM layers used for this ablation experiment.

Channel Combination Mode	Spatial (RGB)	Temporal (OpFlow)
3TGM with 1×1 Conv	**32.5**	16.9
3TGM with summation	29.7	16.8
3TGM with average	32.1	17.0
3TGM with max	31.5	**18.1**

Weighted Branch

An examination of the data presented in Table 5.2 shows that each branch contributes differently to the performance. Thus, it is natural to ask whether using a weighted scheme per branch can increase model performance. We performed several ablation experiments to determine the optimal combination of weights per branch. We weigh each input of the first TGM layer in both the spatial and temporal branches. We changed the values randomly and found the optimal combination of weights. The results are described in Table 5.3.

Table 5.3 Results for different weights for each branch. Proposed #3 is used to produce this result.

Method	mAP (%)
Equal distribution on spatial and temporal branches	37.09
Spatial: 1.25, temporal: 0.95	**37.13**
Spatial: 0.95, temporal: 1.25	37.00

Even though the difference is marginal, the results confirm that each branch contributes to the performance unequally, in agreement with the results presented in Table 5.2.

Possibility to Implement the Previous Approach (META)

It is possible to implement a META module for multi-label multi-class problems for action recognition. In the current implementation, feature vectors for a video are extracted using the Inflated 3D (I3D) network pre-trained on a Kinetics400 dataset. Feature vector extraction is performed for all videos in the MultiTHUMOS dataset and is saved in a numpy array format. These saved numpy files are then used for optimizing the TGM kernels to learn the fine-grained actions.

To enable META for extracting the features from videos is by simply replacing the backbone of the pipeline with META. The rest of the process remains the same. The illustration for this process is depicted in Figure 5.12.

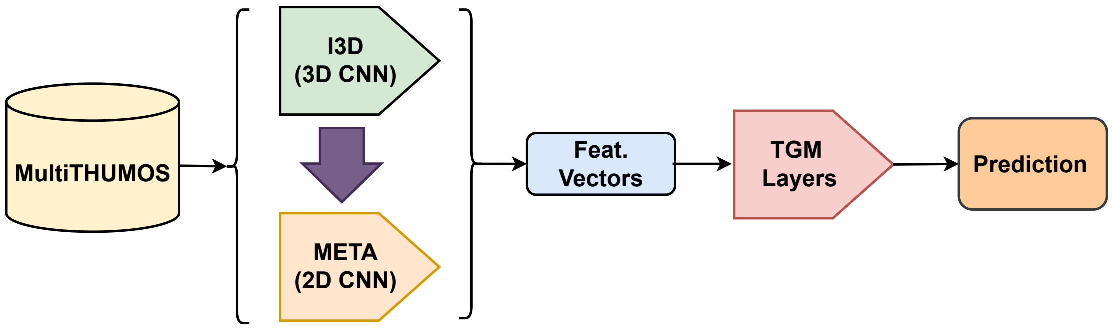


Fig. 5.12 An illustration of switching from I3D to META.

5.5 Summary

In this chapter, we proposed new unified multi-branch neural network models consisting of a sequence of TGM layers that serve as spatial, temporal, and spatio-temporal branches for performing multi-label multi-class classification tasks. Outputs from the spatial and temporal branches are interpolated to form the spatio-temporal branch, with only a few learnable parameters added. We also demonstrated the benefit of using the maximum function inside the TGM layer to combine the input channels. The experimental results have shown that our proposed fusion strategies with the spatio-temporal model learn temporal structure effectively, ultimately improving the activity detection performance and outperforming the baseline and several other designs on the MultiTHUMOS dataset.

Chapter 6

Summary and Future Work

In this dissertation, we have introduced several novel deep learning-based networks to solve video action recognition problems. In particular, we proposed several submodules in a unified network to tackle the existing problems in 2D CNNs in the task of multi-class action classification. Regarding multi-label action classification task, we proposed several novel fusion schemes utilizing temporal gaussian mixture to learn the fine-grained action contained in untrimmed videos.

To conclude this dissertation, we divide this chapter into two sections. First, we summarize the contributions of the presented work in this book. Lastly, we conclude by introducing some prominent directions for future works.

6.1 Summary

In the past chapters, we presented our method to learn the spatiotemporal features for multi-class action classification problems, as discussed in **Chapter 4**). We summarize the contributions in the following:

- **Motion Excitation.** To tackle the issues with the optical flow, which requires vast storage space, we have proposed incorporating the estimation of motion representations in a unified network. By integrating the motion estimation submodule, we endow the model to compute the motion features at the feature level rather than the frame level, as in optical flow. Moreover, we equipped the network with an excitation mechanism, i.e., to suppress less meaningful features while emphasizing the important ones. This way, the model is enforced to focus on the motion-related region more effectively.

- **Temporal Aggregation.** We have introduced local temporal and spatial convolution in subgroups to learn temporal and spatial representations. These subgroups formulate a hierarchical structure with residual connections between adjacent subsets. Its function is to aggregately learn long-range temporal dynamics since the input features realize multiple information exchanges with neighboring frames. Compared to the baseline, this submodule contributes positively to the overall model performance by relaxing the 2D CNN to learn temporal features more faithfully rather than just shifting the kernel partly, as in the baseline implementation.
- **Multi-view Excitation.** We have decomposed spatiotemporal of a video as $T \times H$, $T \times W$, and $W \times W$, rather than traditional $T \times H \times W$ space-time signal, to reduce computation cost during model training or inference. By treating the space-time video frames in this way, we attempted to acquire both efficiency and effectiveness simultaneously. The kernel, which convolves on each view, is shared to control the model complexity at a low level. Apart from factorizing the space-time signal, we let the submodule have the excitation mechanism to achieve a similar impact to that Motion Excitation submodule. Before aggregating all the output features, we multiplied output features from each view with a learnable weight to force all views to contribute proportionally to their performance.
- **Summary of Experiments and Results.** To evaluate the performance of our proposed submodules, we have conducted extensive experiments on three benchmark datasets, including the Something-something v1, Jester, and Moments in Time Mini datasets. Experimental results demonstrated the effectiveness of our proposed methods over the baseline and state-of-the-art methods in terms of top-1 and top-5 accuracies.

Regarding multi-label multi-class action classification problems, several novel deep learning networks are proposed to tackle fine-grained action in a video. Specifically, we have proposed three fusion strategies to learn spatial and temporal representations effectively. We summarize the key contributions of the various fusion strategies (as discussed in **Chapter 5**) in the following paragraph.

- **Spatial and Temporal Fusion.** In multi-label multi-class classification, several actions might co-occur. With the previous approach, it is not easy to distinguish such simultaneous actions. However, our approach is more robust than the baseline since we also jointly consider learning features from spatial and temporal streams for fine-grained action representations to tackle the abovementioned problem. Our approach is also proven to consistently detect an action with a longer duration than the previous approach. In this dissertation, we propose three approaches for learning fine-grained features: (a) two-stream spatial and temporal fusion, (b) three streams of early spatial and temporal fusion, and (c) three-stream of multi-level spatial and temporal.

- **Summary of Experiments and Results.** We have validated the effectiveness of our approaches on a MultiTHUMOS dataset. Experimental results demonstrated the effectiveness of our proposed features over the baseline and other related works.

6.2 Future Work

In the future, we plan to incorporate the Vision Transformer-based network into our work. Vision Transformers are rapidly starting to dominate many applications in Computer Vision, including action recognition tasks. Transformer-based techniques dominate the leaderboard of primary datasets for action recognition. As discussed in the previous chapter, one of the strengths of a Transformer-based network is the ability to capture global information from earlier layers. According to our research, the model's capacity to effectively learn motion representation is a prerequisite for successful performance on action recognition tasks. For efficient end-to-end learning, these motion representation blocks must be included in the network. We contend that adding a motion encoder to the Vision Transformer may improve its modeling capabilities.

Related Publications

Articles in International Journals:

- Yuri Yudhaswana Joefrie and Masaki Aono: “Multi-label Multi-class Action Recognition with Deep Spatio-Temporal Layers based on Temporal Gaussian Mixtures”, IEEE Access, vol. 8, pp. 173566–173575.2020.
- Yuri Yudhaswana Joefrie and Masaki Aono: “Video Action Recognition using Motion and Multi-view Excitation with Temporal Aggregation”, MDPI Entropy, vol. 24, 1663. 2022.

Articles in International Conferences:

Yuri Yudhaswana Joefrie and Masaki Aono: “Action Recognition by Composite Deep Learning Architecture I3D-DenseLSTM”, In the 2019 International Conference on Advanced Informatics: Concepts, Theory, and Application, (ICAICTA), pp. 91–96, September 20–22nd, 2019, (IEEE).

References

[1] Abdelbaky, A. and Aly, S. (2021). Two-stream spatiotemporal feature fusion for human action recognition. *The Visual Computer*, 37(7):1821–1835.

[2] Alexandrie, G. (2017). Surveillance cameras and crime: a review of randomized and natural experiments. *Journal of Scandinavian Studies in Criminology and Crime Prevention*, 18:1–13.

[3] Arandjelović, R. and Zisserman, A. (2012). Three things everyone should know to improve object retrieval. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2911–2918.

[4] Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lucic, M., and Schmid, C. (2021). ViViT: A Video Vision Transformer. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6816–6826.

[5] Bertasius, G., Wang, H., and Torresani, L. (2021). Is Space-Time Attention All You Need for Video Understanding? In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 813–824. PMLR.

[6] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[7] Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-based systems*, 46:109–132.

[8] Brezovský, M., Sopiak, D., and Oravec, M. (2018). Action recognition by 3d convolutional network. *Proceedings Elmar - International Symposium Electronics in Marine*, 2018-Septe:71–74.

[9] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370.

[10] Cai, D. (2018). Trimmed Event Recognition (Moments in Time): Submission to ActivityNet Challenge 2018. Technical report.

[11] Carreira, J. and Zisserman, A. (2017). Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733. IEEE.

[12] Dai, W., Chen, Y., Huang, C., Gao, M.-k., and Zhang, X. (2019). Two-Stream Convolution Neural Network with Video-stream for Action Recognition. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

[13] Dave, A., Russakovsky, O., and Ramanan, D. (2017). Predictive-corrective networks for action detection. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 2067–2076. Institute of Electrical and Electronics Engineers Inc.

[14] Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, and Li Fei-Fei (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE.

[15] Dennis, D., Pabbaraju, C., Simhadri, H. V., and Jain, P. (2018). Multiple Instance Learning for Efficient Sequential Data Classification on Resource-constrained Devices. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31, page 12pp. Curran Associates, Inc.

[16] Diba, A., Fayyaz, M., Sharma, V., Karami, A. H., Arzani, M. M., Yousefzadeh, R., and Gool, L. V. (2017). Temporal 3D ConvNets: New Architecture and Transfer Learning for Video Classification. *CoRR*, abs/1711.0:9pp.

[17] Feichtenhofer, C., Fan, H., Malik, J., and He, K. (2019). SlowFast Networks for Video Recognition. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 6201–6210. IEEE.

[18] Feichtenhofer, C., Pinz, A., and Zisserman, A. (2016). Convolutional Two-Stream Network Fusion for Video Action Recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 1933–1941. IEEE Computer Society.

[19] Fukushima, K. (2004). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202.

[20] Gandomi, A. and Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 35(2):137–144.

[21] Gao, S.-H., Cheng, M.-M., Zhao, K., Zhang, X.-Y., Yang, M.-H., and Torr, P. (2019). Res2Net: A New Multi-scale Backbone Architecture. *IEEE transactions on pattern analysis and machine intelligence*, 43(2):652–662.

[22] Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning.* MIT Press, Cambridge, MA, USA.

[23] Goyal, R., Ebrahimi Kahou, S., Michalski, V., Materzynska, J., Westphal, S., Kim, H., Haenel, V., Fruend, I., Yianilos, P., Mueller-Freitag, M., Hoppe, F., Thurau, C., Bax, I., and Memisevic, R. (2017). The "Something Something" Video Database for Learning and Evaluating Visual Common Sense. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, page 9pp.

[24] Guan, S. and Li, H. (2018). SYSU iSEE submission to Moments in Time Challenge 2018. Technical report, School Data Comput. Sci., Sun Yat-Sen Univ., Guangzhou, China, Tech. Rep., page 3pp.

[25] Gupta, C., Suggala, A. S., Goyal, A., Simhadri, H. V., Paranjape, B., Kumar, A., Goyal, S., Udupa, R., Varma, M., and Jain, P. (2017). {P}roto{NN}: Compressed and Accurate k{NN} for Resource-scarce Devices. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1331–1340. PMLR.

[26] Hao, Y., Wang, S., Cao, P., Gao, X., Xu, T., Wu, J., and He, X. (2022). Attention in Attention: Modeling Context Correlation for Efficient Video Classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(10):7120–7132.

[27] Hara, K., Kataoka, H., and Satoh, Y. (2017). Learning spatio-Temporal features with 3D residual networks for action recognition. In *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017*, volume 2018-Janua, pages 3154–3160. Institute of Electrical and Electronics Engineers Inc.

[28] Hara, K., Kataoka, H., and Satoh, Y. (2018). Towards Good Practice for Action Recognition with Spatiotemporal 3D Convolutions. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2516–2521.

[29] Harrendorf, S., Heiskanen, M., and Malby, S. (2010). International Statistics on Crime and Justice. page 178pp.

[30] He, D., Zhou, Z., Gan, C., Li, F., Liu, X., Li, Y., Wang, L., and Wen, S. (2019). StNet: Local and Global Spatial-Temporal Modeling for Action Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:8401–8408.

[31] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:770–778.

[32] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

[33] Hu, J., Shen, L., Albanie, S., Sun, G., and Wu, E. (2017). Squeeze-and-Excitation Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(8):2011–2023.

[34] Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF Models for Sequence Tagging. page 10pp.

[35] Hubel, D. H. and Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 195:215–243.

[36] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2017). FlowNet 2.0: Evolution of optical flow estimation with deep networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:1647–1655.

[37] Isinkaye, F. O., Folajimi, Y. O., and Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273.

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

[39] Ji, S., Xu, W., Yang, M., and Yu, K. (2013a). 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231.

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

[41] Jiang, B., Wang, M., Gan, W., Wu, W., and Yan, J. (2019). STM: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 2000–2009. Institute of Electrical and Electronics Engineers Inc.

[42] Joefrie, Y. Y. and Aono, M. (2019). Action Recognition by Composite Deep Learning Architecture I3D-DenseLSTM. In *2019 International Conference of Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, pages 1–6. IEEE.

[43] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale Video Classification with Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 8pp.

[44] Khalid, M. U. and Yu, J. (2018). Multi-Modal Three-Stream Network for Action Recognition. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3210–3215.

[45] Kong Yu, , and Fu, Y. (2022). Human Action Recognition and Prediction: A Survey. *International Journal of Computer Vision*, 130(5):1366–1401.

[46] Kumar, A., Goyal, S., and Varma, M. (2017). Resource-efficient Machine Learning in 2 {KB} {RAM} for the Internet of Things. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1935–1944. PMLR.

[47] La Vigne, N., Lowry, S., Markman, J., and Dwyer, A. (2011). Evaluating the Use of Public Surveillance Cameras for Crime Control and Prevention- A Summary. Technical report.

[48] Lee, M., Lee, S., Son, S., Park, G., and Kwak, N. (2018). Motion Feature Network: Fixed Motion Filter for Action Recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11214 LNCS:392–408.

[49] Li, C., Hou, Z., Chen, J., Bu, Y., Zhou, J., Zhong, Q., Xie, D., and Pu, S. (2018a). Team DEEP-HRI Moments in Time Challenge 2018 Technical Report. page 3pp.

[50] Li, K., Wang, Y., Gao, P., Song, G., Liu, Y., Li, H., and Qiao, Y. (2022). UniFormer: Unified Transformer for Efficient Spatial-Temporal Representation Learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, page 19pp. OpenReview.net.

[51] Li, Q., Peng, Q., and Yan, C. (2018b). Multiple VLAD Encoding of CNNs for Image Classification. *Computing in Science & Engineering*, 20(2):52–63.

[52] Li, Y., Ji, B., Shi, X., Zhang, J., Kang, B., and Wang, L. (2020). TEA: Temporal Excitation and Aggregation for Action Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 906–915.

[53] Lin, J., Gan, C., and Han, S. (2018). TSM: Temporal Shift Module for Efficient Video Understanding. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:7082–7092.

[54] Lv, Y., Zheng, H., and Zhang, W. (2018). Multi-level Three-Stream Convolutional Networks for Video-Based Action Recognition. In Lai, J.-H., Liu, C.-L., Chen, X., Zhou, J., Tan, T., Zheng, N., and Zha, H., editors, *Pattern Recognition and Computer Vision*, pages 237–249, Cham. Springer International Publishing.

[55] Materzynska, J., Berger, G., Bax, I., and Memisevic, R. (2019). The Jester Dataset: A Large-Scale Video Dataset of Human Gestures. In *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*, pages 2874–2882. IEEE.

[56] Meng, W., Gu, Z., Zhang, M., and Wu, Z. (2017). Two-Bit Networks for Deep Learning on Resource-Constrained Embedded Devices. *CoRR*, abs/1701.0:2pp.

[57] Monfort, M., Andonian, A., Zhou, B., Ramakrishnan, K., Bargal, S. A., Yan, T., Brown, L., Fan, Q., Gutfreund, D., Vondrick, C., and Oliva, A. (2020). Moments in Time Dataset: One Million Videos for Event Understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(2):502–508.

[58] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[59] Perronnin, F., Liu, Y., Sánchez, J., and Poirier, H. (2010). Large-scale image retrieval with compressed Fisher vectors. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3384–3391.

[60] Piergiovanni, A. J. and Ryoo, M. S. (2019). Temporal Gaussian Mixture Layer for Videos. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5152–5161. PMLR.

[61] Piza, E. L., Welsh, B. C., Farrington, D. P., and Thomas, A. L. (2019). CCTV surveillance for crime prevention. *Criminology & Public Policy*, 18(1):135–159.

[62] Qiu, Z., Yao, T., and Mei, T. (2017). Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 5534–5542. Institute of Electrical and Electronics Engineers Inc.

[63] Qiu, Z., Yao, T., Ngo, C.-W., Tian, X., and Mei, T. (2019). Learning Spatio-Temporal Representation With Local and Global Diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12056–12065. Computer Vision Foundation / IEEE.

[64] Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., and Dosovitskiy, A. (2021). Do Vision Transformers See Like Convolutional Neural Networks? In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 12116–12128.

[65] Ramezani, M. and Yaghmaee, F. (2016). A review on human action analysis in videos for retrieval applications. *Artif. Intell. Rev.*, 46(4):485–514.

[66] Rumelhart, D. E. and McClelland, J. L. (1987). Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, pages 318–362.

[67] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2014). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

[68] Sevilla-Lara, L., Liao, Y., Güney, F., Jampani, V., Geiger, A., and Black, M. J. (2019). On the Integration of Optical Flow and Action Recognition. In Brox, T., Bruhn, A., and Fritz, M., editors, *Pattern Recognition*, pages 281–297, Cham. Springer International Publishing.

[69] Sigurdsson, G. A., Varol, G., Wang, X., Farhadi, A., Laptev, I., and Gupta, A. (2016). Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision - {ECCV} 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part {I}*, volume 9905 of *Lecture Notes in Computer Science*, pages 510–526. Springer.

[70] Simonyan, K. and Zisserman, A. (2014). Two-Stream Convolutional Networks for Action Recognition in Videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pages 568–576, Cambridge, MA, USA. MIT Press.

[71] Stan, J., Muhlenbach, F., and Largeron, C. (2014). Recommender Systems Using Social Network Analysis: Challenges and Future Trends. In *Encyclopedia of Social Network Analysis and Mining*, pages 1522–1532.

[72] Stroud, J. C., Ross, D. A., Sun, C., Deng, J., and Sukthankar, R. (2018). D3D: Distilled 3D Networks for Video Action Recognition. *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020*, pages 614–623.

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

[74] Sun, L., Jia, K., Yeung, D.-Y., and Shi, B. E. (2015b). Human Action Recognition Using Factorized Spatio-Temporal Convolutional Networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4597–4605. IEEE.

[75] Tian, Y., Yan, Y., Zhai, G., Guo, G., and Gao, Z. (2022). EAN: Event Adaptive Network for Enhanced Action Recognition. *Int. J. Comput. Vis.*, 130(10):2453–2471.

[76] Tran, D., Bourdev, L. D., Fergus, R., Torresani, L., and Paluri, M. (2015). Learning Spatiotemporal Features with 3D Convolutional Networks. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 4489–4497. IEEE Computer Society.

[77] Tran, D., Wang, H., Torresani, L., Ray, J., Lecun, Y., and Paluri, M. (2017). A Closer Look at Spatiotemporal Convolutions for Action Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 6450–6459.

[78] Truong, T.-D., Bui, Q.-H., Duong, C. N., Seo, H.-S., Phung, S. L., Li, X., and Luu, K. (2022). DirecFormer: A Directed Attention in Transformer Approach to Robust Action Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 19998–20008. IEEE.

[79] Uchida, Y. and Sakazawa, S. (2013). Image Retrieval with Fisher Vectors of Binary Features. In *2nd IAPR Asian Conference on Pattern Recognition, ACPR 2013, Naha, Japan, November 5-8, 2013*, pages 23–28. IEEE.

[80] Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., and Baik, S. W. (2018). Action Recognition in Video Sequences using Deep Bi-Directional LSTM With CNN Features. *IEEE Access*, 6:1155–1166.

[81] Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. (2013). Dense Trajectories and Motion Boundary Descriptors for Action Recognition. *Int. J. Comput. Vis.*, 103(1):60–79.

[82] Wang, H., Oneata, D., Verbeek, J., and Schmid, C. (2016a). A Robust and Efficient Video Representation for Action Recognition. *Int. J. Comput. Vision*, 119(3):219–238.

[83] Wang, H. and Schmid, C. (2013). Action Recognition with Improved Trajectories. In *2013 IEEE International Conference on Computer Vision*, pages 3551–3558. IEEE.

[84] Wang, L., Li, W., and Van Gool, L. (2017). Appearance-and-Relation Networks for Video Classification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1430–1439.

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

[86] Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Van Gool, L. (2016b). Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 20–36, Cham. Springer International Publishing.

[87] Wang, X. and Gupta, A. (2018). Videos as Space-Time Region Graphs. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part V*, volume 11209 of *Lecture Notes in Computer Science*, pages 413–431. Springer.

[88] Wang, Z., She, Q., and Smolic, A. (2021). ACTION-Net: Multipath Excitation for Action Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 13209–13218.

[89] Wray, M., Doughty, H., and Damen, D. (2021). On Semantic Similarity in Video Retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 3650–3660. Computer Vision Foundation / IEEE.

[90] Xie, B., Qin, J., Xiang, X., Li, H., and Pan, L. (2018a). An Image Retrieval Algorithm Based on Gist and Sift Features. *Int. J. Netw. Secur.*, 20:609–616.

[91] Xie, S., Sun, C., Huang, J., Tu, Z., and Murphy, K. (2018b). Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*, volume 11219 of *Lecture Notes in Computer Science*, pages 318–335. Springer.

[92] Yang, C., Xu, Y., Shi, J., Dai, B., and Zhou, B. (2020). Temporal Pyramid Network for Action Recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 588–597. Computer Vision Foundation / IEEE.

[93] Yang, L. and Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316.

[94] Yang, Q., Lu, T., and Zhou, H. (2022). A Spatio-Temporal Motion Network for Action Recognition Based on Spatial Attention. *Entropy*, 24(3):19pp.

[95] Yeung, S., Russakovsky, O., Jin, N., Andriluka, M., Mori, G., and Fei-Fei, L. (2018). Every Moment Counts: Dense Detailed Labeling of Actions in Complex Videos. *Int. J. Comput. Vis.*, 126(2-4):375–389.

[96] You, J., Shi, P., and Bao, X. (2018). Multi-stream I3D network for fine-grained action recognition. In *Proceedings of 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference, ITOEC 2018*, pages 611–614. Institute of Electrical and Electronics Engineers Inc.

[97] Zach, C., Pock, T., and Bischof, H. (2007). A Duality Based Approach for Realtime TV-L 1 Optical Flow. In *Pattern Recognition*, pages 214–223. Springer Berlin Heidelberg, Berlin, Heidelberg.

[98] Zhang, Y. (2022). MEST: An Action Recognition Network with Motion Encoder and Spatio-Temporal Module. *Sensors*, 22(17):17pp.

[99] Zhao, Y., Xiong, Y., Wang, L., Wu, Z., Tang, X., and Lin, D. (2017). Temporal Action Detection with Structured Segment Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 2933–2942. Institute of Electrical and Electronics Engineers Inc.

[100] Zhou, B., Andonian, A., Oliva, A., and Torralba, A. (2018). Temporal Relational Reasoning in Videos. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I*, volume 11205 of *Lecture Notes in Computer Science*, pages 831–846. Springer.

[101] Zolfaghari, M., Singh, K., and Brox, T. (2018). ECO: Efficient Convolutional Network for Online Video Understanding. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11206 LNCS:713–730.