EXPLORING DEEP SPATIO-TEMPORAL FUSION ARCHITECTURES TOWARDS LATE TEMPORAL MODELING OF HUMAN ACTION RECOGNITION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MUHAMMET ESAT KALFAOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

AUGUST 2020

Approval of the thesis:

EXPLORING DEEP SPATIO-TEMPORAL FUSION ARCHITECTURES TOWARDS LATE TEMPORAL MODELING OF HUMAN ACTION RECOGNITION

submitted by **MUHAMMET ESAT KALFAOĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İlkay Ulusoy _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. A. Aydın Alatan _____
Supervisor, **Electrical and Electronics Engineering, METU**

Assoc. Prof. Dr. Sinan Kalkan _____
Co-supervisor, **Computer Engineering, METU**

Examining Committee Members:

Assoc. Prof. Dr. Fatih Kamışlı
Electrical and Electronics Engineering, METU _____

Prof. Dr. A. Aydın Alatan
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Nazlı İkizler Cimbiş
Computer Engineering, Hacettepe University _____

Assist. Prof. Dr. Elif Sürer
Multimedia Informatics, METU _____

Assist. Prof. Dr. Emre Akbaş
Computer Engineering, METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MUHAMMET ESAT KALFAOĞLU

Signature :

ABSTRACT

EXPLORING DEEP SPATIO-TEMPORAL FUSION ARCHITECTURES TOWARDS LATE TEMPORAL MODELING OF HUMAN ACTION RECOGNITION

Kalfaoğlu, Muhammet Esat
M.S., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. A. Aydın Alatan
Co-Supervisor : Assoc. Prof. Dr. Sinan Kalkan

August 2020, 126 pages

Visual action recognition (AR) is the problem of identifying the labels of activities that occur in a video. In this thesis, different spatio-temporal representations are analyzed and the factors making these representations better suited for AR are determined. To be specific, three main concepts are analyzed in this thesis study which are the effects of different architectural selections, the input modalities (RGB, optical flow, human pose), and temporal modeling concepts. Additionally, the joint utilization of BERT-based late temporal modeling with 3D CNN architectures is proposed and a novel distillation concept is recommended within this approach.

Firstly, for architectural analysis, both 2D and 3D CNN structures are considered. For 3D CNN architectures, the effects of clip length, input spatial resolution, group convolution, and separable 3D convolution are analyzed. During this analysis, popular 3D CNN architectures for AR, such as MFNET, SlowFast Networks, R(2+1)D networks, I3D, MARS networks (knowledge distillation), and various ResNet architectures are all considered. Temporal shift modules are also investigated as an extension to 2D CNN architectures.

For input modality analysis, popular two-stream architectures (RGB+Flow) are analyzed within both 2D and 3D CNN architectures. Moreover, as an extension to RGB

and flow modalities, pose input modality is utilized with a different approach from the literature and studied within the 2D CNN architectures in this thesis.

For the temporal modeling analysis, various techniques are analyzed such as average pooling, LSTM, convolutional GRU, BERT, and non-local blocks within 2D CNN architectures.

As a novel extension, conventional 3D convolutions are combined with late temporal modeling for AR. The popular temporal global average pooling layer (TGAP) at the end of 3D convolutional architecture is replaced with the recent Bidirectional Encoder Representations from Transformers (BERT) layer in order to better exploit the attention mechanism of BERT. Such a replacement is shown to improve the performances of many popular 3D convolution architectures, including ResNeXt, I3D, SlowFast, and R(2+1)D. The-state-of-the-art performances are obtained on both HMDB51 and UCF101 datasets with 85.10% and 98.69% Top-1 accuracy, respectively. Finally, a novel knowledge distillation concept is proposed using a 3D-BERT architecture that yields quite promising performances.

Keywords: Activity Recognition, Action Recognition, Temporal Modeling, 3D Convolution, Two-Stream Networks, Spatio-Temporal Features

ÖZ

İNSAN AKTİVİTELERİNİ TANIMA İÇİN DERİN UZAM-ZAMANSAL FÜZYON MİMARİLERİN GEÇ ZAMANSAL MODELLEMEYE YÖNELİK İNCELENMESİ

Kalfaoğlu, Muhammet Esat
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. A. Aydın Alatan
Ortak Tez Yöneticisi : Doç. Dr. Sinan Kalkan

Ağustos 2020 , 126 sayfa

Görsel eylem tanıma (ET), bir videoda meydana gelen eylemlerin ne olduğunu tanımlama problemidir. Bu tezde, farklı uzam-zamansal yapılar analiz edilmiş ve bu gösterimleri ET için daha uygun hale getiren faktörler belirlenmiştir. Spesifik olmak gerekirse, bu tez çalışmasında farklı mimari seçimlerin, girdi modalitelerinin (RGB, optik akış, insan pozu) ve zamansal modelleme kavramlarının etkileri üç ana kavram olarak ele alınmıştır. Ek olarak, BERT tabanlı geç zamansal modellemenin 3D CNN mimarileri ile ortak kullanımı önerilmiş ve bu yaklaşım içinde yeni bilgi damıtma kavramı önerilmiştir.

Mimari analiz için hem 2D hem de 3D Evrişimsel Sinir Ağları (CNN) dikkate alınır. 3D CNN mimarileri için girdi klip uzunluğu, girdi uzamsal çözünürlüğü, grup evrişimi ve ayrılabilir 3D evrişim mimarilerinin etkileri analiz edilir. Bu analiz sırasında, MFNET, SlowFast Networks, R(2 + 1)D ağları, I3D, MARS ağları (bilgi damıtma) ve çeşitli ResNet mimarileri gibi AR için popüler 3D CNN mimarilerinin tümü dikkate alınır. Zamansal kayma modülleri ayrıca 2D CNN mimarilerinin bir uzantısı olarak incelenir.

Girdi modalite analizi için, popüler iki kanallı mimariler (RGB + optik akış) hem 2D hem de 3D CNN mimarileri içinde analiz edilir. Ayrıca, RGB ve optik akış moda-

litelerinin bir uzantısı olarak, poz girdi modalitesi literatürden farklı bir yaklaşımla kullanılmıştır ve bu tezde 2D CNN mimarileri dahilinde incelenmiştir.

Zamansal modelleme analizi için, 2D CNN mimarileri içinde ortalama havuzlama, LSTM, evrişimli GRU, BERT ve Yerel Olmayan blok yapıları gibi çeşitli teknikler analiz edilir.

Yeni bir öneri olarak, bu çalışmada, ET problemi için 3D evrişim mimarilerinin geç zamansal modelleme ile birleştirilmesi sunulmuştur. Bu amaçla 3D evrişimsel mimarilerinin sonundaki geleneksel zamansal ortalama havuz katmanı (TGAP) Transformatörlerden Çift Yönlü Enkoder Temsilleri (BERT) katmanıyla değiştirilmiş ve BERT'nin ilgi mekanizmasıyla daha iyi bir geç zamansal modelleme amaçlanmıştır. Bu değiştirmenin, ResNeXt, I3D, SlowFast ve R(2 + 1)D gibi eylem tanıma için birçok popüler 3D evrişim mimarisinin performansını geliştirdiği gösterilmiştir. Ayrıca, HMDB51 ve UCF101 veri kümelerinde sırasıyla 85.10% ve 98.69% top-1 doğruluğu ile literatürdeki en gelişmiş sonuçlar sunulmuştur. Ayrıca, 3D-BERT mimarisi üzerinden bir bilgi damıtma yapısı önerilmiş ve analiz edilmiştir.

Anahtar Kelimeler: Aktivite Tespiti, Zamansal Modelleme, 3D Evrişim, İki Kanallı Ağlar, Uzamsal-Zamansal Öznitelikler

To my beloved family and friends

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my supervisor Prof. Dr. A. Aydın Alatan for his continuous support, guidance, patience, encouragement in the path of creation of this thesis. I learnt a lot not only in the perspective of research but also the ethical concerns and the administration. I thank him for his contribution to my scientific vision, my writing skills and for the great lab environment that he provides us. I am very grateful for his immediate responses when I need some help in anything I have asked for and the great positive communication and understanding which he provides for me.

Secondly, I want to thank the greatest support to my co-advisor Assoc. Prof. Dr. Sinan Kalkan for his contribution to my project and my thesis work. I learn to look at the problems from a different perspective, and I improved my writing skills very much thanks to his contribution. I am very grateful for his effort to create creative ideas and his great communication.

I also would like to thank to my friends from Center for Image Analysis (OGAM). Special thanks to Alp Eren Sarı for his friendship and all kinds of support and help. We have shared lots of great memories together. He is really a genuine friend for me. It was a pleasure being in the same project with him. I won't forget the good memories of England with him. Thanks to Oğul Can for directing my studies towards BERT architecture which has strong impact on this thesis and for his good friendship. Thanks to Dr. Alper Koz, Mustafa Ergül, Yeti Ziya Gürbüz, Dr. Gökhan Koray Gültekin, Dr Kutalmış Ince and İlker Gürcan for their advices about both the academy and the personal life. Thanks to Ece Selin Böncü for sharing valuable moments with me. She has a great personality and I wont forget the day of Bahri Abi with her and Oğul. Thanks to Ayberk Aydın for his genuine friendship, sharing his profound knowledge about the literature with us and his child stuff. It was a pleasure to work back to back with him. Thanks to Ihsan Emre Üstün for the cherries that he brings from his hometown. He is a very good friend and he has a very kind personality. Thanks to Aziz Berkay Yeşilyurt for his help and his friendship. Sometimes, his assistance can be lifesaving. Thanks to Aybüke Erol for her genuine friendship. She is the one who makes us not forget Aşk-ı Memnu and Titanic. She is the one with very creative questions and she is very good at multi-tasking. Thanks to İzlen Geneci for her good friendship. Sometimes, I lose the track of time while chatting with her. Thanks to Gamze Sever for her friendship and help. I have asked her help many times about the administrative affairs and she always helps me without reluctance. Thanks to Mustafa Kütük and Emre Can Kaya for their friendship. They help me a

lot in my adaptation to the OGAM environment. Thanks to Ufuk Efe, Ahmet Arslan, Mert Alp Öcalp, Can Çağlayan Çakırgöz, Aybora Köksal for their valuable times and friendship. I also should not forget to thank Bahar Şengün and Havva Oğuz for their sincere help about administrative affairs.

I also would like to present my special thanks to my family who always supports me in my decisions. The biggest portion of the thank belongs to my beloved mom. She is the hidden hero of this thesis. She always take care of me whenever and wherever I need help without considering herself. Secondly, I want to present my thanks to my father who has significant guidance in my decisions and always gives the freedom in my choices. Thanks to my sister and brother-in-law who let me stay in their home and give me a ride when I needed. Thanks to my brother and sister-in-law for being guarantors for my TÜBİTAK 2210/A scholarship and their support.

I also would like to thank Assoc. Prof. Dr. İpek Gürsel Dino for giving a chance to work in a project (SISER) which contributes the works in my thesis work. Thanks to her academic vision, I have lots of academic outputs. It is also a pleasure to work with Şahin Akın and Orçun Koral İşeri in the same project. They are really hardworking and problem-solver people.

I also want present my special thanks to my friends Fatih Çağatay Akyön, Ahmet Safa Öztürk, İbrahim Tanrıöver, İbrahim Kurban Özaslan, Botan Yıldırım and Tuğrul Görgülü for their sincere friendship and their guidance about academic life. It was a pleasure to share the same dormitory room with Tuğrul Görgülü.

This work is supported by an Institutional Links grant under the Newton-Katip Çelebi partnership, Grant No. 217M519 by the Scientific and Technological Research Council of Turkey (TÜBİTAK) and ID [352335596] by British Council, UK.

This work is supported by TÜBİTAK within the scope of 2210/A scholarship.

The numerical calculations reported in this paper were partially performed at TÜBİTAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources).

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Activity recognition or action recognition (AR) is the task of recognition of predefined actions or activities from "short" video clips ("short" duration is typically assumed to be between 2 to 15 seconds). There are two important information sources in a video clip for the action recognition task: spatial information and temporal information.

The first information source is the spatial information, which can be defined as what someone can obtain from a still image or a single frame. In another definition, it is defined as the static information existing in a single frame, such as entities, objects, and context. For example, the action 'swimming' can be identified from a still image, if a person in a water body can be recognized.

The other major information source, which is quite critical for AR, is the temporal information. This information source is based on the relationship between the frames of a video sequence. For example, if a person is in a water body, it is hard to capture the difference between the *breast crawl* and *breast stroke*, which are two different swimming styles.

Another typical example in which the temporal information is crucial is discrimination between opening and closing actions on a window (or door). Assume that only one frame is extracted from the video sequence consisting of opening and closing window actions (See Figure 1.1). It is quite hard to recognize whether it belongs to the opening or closing actions. For this purpose, there is an extra requirement to understand where a person moves or where a window rotates.

(a) A frame sample from the clip of *closing* window action.

(b) A frame sample from a the clip of *opening* window action.

Figure 1.1: The visual demonstration of the significance of temporal information for the distinction of reverse actions.

Another challenge is distinguishing actions which are similar but evolving at different speeds, such as walking and running. As a consequence, in order to perform reliable action recognition, the representations that are extracted from a video clip should model both spatial and temporal information simultaneously and the representations should be aware and invariant of the evolution speed of the performed action.

In any real-world application of AR, not only the classification performance but also the computational complexity and memory utilization are two important factors. Tackling such a difficult problem using conventional machine learning techniques requires huge computational complexity and memory utilization. Therefore, the ultimate goal of our AR research is obtaining good performance while considering the computational complexity and memory utilization.

1.1 Applications of Action Recognition

Action recognition (AR) can be utilized in many different areas and applications, such as video retrieval, surveillance systems, human-computer interaction, robot perception, and sign language.

Video retrieval is the problem of finding similar video clips to the given input video clip. It is an important concept in dealing with huge video archives. Another im-

portant area for AR is surveillance. In video surveillance, detecting any problematic action as early as possible can be crucial for preventing advert events, such as robbery or fights. Human-machine interaction is another domain where AR is crucial. For instance, in virtual reality or computer games, detecting gestures can be useful. Similarly, AR can be utilized in automatic sign language recognition.

Another domain where AR can be utilized is robot perception. For robots to be able to freely interact with humans, AR should be an indispensable part of robot perception, since the first step of taking proper actions is to comprehend the situation. For example, consider an example of robot surgery. This requires having an understanding of the surgeon to make convenient interventions for the surgery. Another example can be physiotherapy exercises where the correctness of an exercise can be evaluated by AR systems.

1.2 Scope and Contributions of the Thesis

This thesis covers various 2D and 3D CNN architectures for the AR problem. 3D CNN architectures have become quite popular due to their success in the AR literature. However, 3D CNN architectures have higher computational complexity and more parameters than their 2D CNN counterparts. Therefore, there are studies which try to decrease the computational burden of 3D CNNs, while preserving their accuracy [64, 76, 6, 24]. For this aim, this thesis makes an in-depth analysis of 2D and 3D CNN architectures with their performances, the number of parameters, and the number of operations.

A second focus of the thesis is regarding the input modality. There are studies that show the positive impact of utilizing different input modalities jointly in the performance of the AR problem. [57, 17, 5, 48, 71, 49, 8, 64, 76, 68, 12, 13, 78, 7, 18]. For this aim, the effect of RGB videos, estimated motion vectors (i.e. optical flow field) of the points in the scene, automatically determined human pose positions and their joint utilization (two-stream and three-stream) are analyzed in this thesis.

Moreover, the thesis analyzes late temporal modeling of the learned spatio-temporal representations for both 2D and 3D CNN architectures. Within this context, a novel

BERT-based late temporal modeling on 3D CNN architectures is proposed in this thesis.

Finally, various concepts are examined for the AR problem, such as the effect of the distillation concept in which one network transfers its knowledge to the other network, scale of the pre-training dataset, clip length of 3D CNNs, feature fusion. Additionally, knowledge distillation for AR is extended by using BERT architectures and this novel method is analyzed by using 3D CNN architectures.

The work presented in the Chapter 4 of the thesis has been disseminated in a paper [30]:
E. Kalfaoglu, S. Kalkan, A. Alatan, “Late Temporal Modeling in 3D CNN Architectures with BERT for Action Recognition", ECCV2020 2nd Workshop on Video Turing Test: Toward Human-Level Video Story Understanding, 2020.

1.3 Outline of the Thesis

In this thesis, the related work from the literature is presented in Chapter 2. In order to better assess the advantages of the related work, an experimental analysis of the literature is presented in Chapter 3. In this chapter, datasets, implementation details, experimental results related with 2D CNN architectures and experiments related with 3D CNN architectures are given in Sections 3.1, 3.2, 3.3 and 3.4 respectively. Next, a proposed method based on BERT which is applied on 3D CNN architectures is presented in Chapter 4, where BERT is implemented for late temporal modeling on top of the 3D CNN architectures. After this chapter, another novel method for BERT-based knowledge distillation is analyzed in Chapter 5. Finally, the thesis is concluded with Chapter 6 where the summary and concluding remarks of the thesis are presented. Some complementary concepts, such as the utilized optical flow and pose estimation techniques or the experimental details are all presented in the Appendix section of the thesis.

CHAPTER 2

RELATED WORK

In this chapter, the literature related to the scope of this thesis is covered. Firstly, an overview of action recognition literature is presented. Following this overview, the prominent deep-learning-based methods and concepts from the literature are presented in detail in this chapter.

2.1 An Overview of Action Recognition (AR) Literature

In this section, a general overview of the AR literature is presented without giving detailed explanations. This section is divided into two main parts, which are pre-deep learning AR literature and deep learning AR literature, while the main focus of this thesis is more on the latter. Therefore, the analysis of deep learning AR literature will be more elaborate.

2.1.1 Pre-deep-learning AR literature

Before the introduction of deep learning, action recognition methods depended on the extracted hand-crafted features. The general block diagram of pre-deep-learning methods is presented in Figure 2.1. The pre-deep-learning methods for AR consist of three main stages: spatio-temporal interest point detection, creations of spatio-temporal description for detected interest points, and the encoding of interest point descriptions.

As mentioned before, both the spatial information and temporal information are cru-

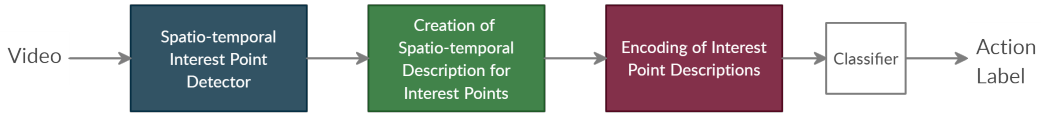


Figure 2.1: General block diagram of pre-deep learning methods

cial for action recognition problem. Therefore, differently from an image classification task, there is a need to process also the temporal information for action recognition. From that perspective, the keyword *spatio-temporal* should be highlighted for both interest point detection and description.

The spatio-temporal interest point detectors can be categorized into two from the perspective of providing temporal characteristics to spatial detectors. These are *spatio-temporal detectors* which the extension of spatial detectors with the temporal dimension [37, 10, 73, 38] and *trajectory-based detectors* which are the temporal trajectories of spatial or spatio-temporal interest points. The extension of spatial detectors with the temporal dimension means that detectors aim to find three-dimensional blobs or corners in a video. In temporal trajectories, the spatial interest points which are detected from a single frame or spatio-temporal interest points, such as Harris3D are tracked with KLT, SIFT-matching and optical flow algorithms [45, 46, 29, 66].

The spatio-temporal descriptors can also be categorized into two from providing temporal characteristics to spatial descriptors. These are the extension of spatial descriptions with the temporal dimension [54, 35, 73] and pooling of spatial descriptions across temporal dimension [66, 38]. For instance, pooling can be a concatenation. The trajectories themselves can also be descriptors [66, 45].

Interest point descriptions are obtained by aggregating local descriptions of the detected interest points into fixed-sized video-level features. This can be achieved via Visual Bag of Words (VBoW), or Fisher Vector Representations [53] or Vector of Locally Aggregated Descriptors (VLAD) vector. Visual Bag of Words can be implemented via K-means or Gaussian Mixture Models (GMM). The difference between K-means and GMM is that K-means is the hard assignment of the feature vector to the clusters and assumes that one feature only belongs to the one cluster of K-means while GMM is the soft assignment of the feature vector to the clusters and defines

probabilities for the assignments of the clusters. Fisher Vectors depends on the idea that score function can be defined as the gradient of probability modeling and describes how the parameters of the model should be modified to represent the model better. For this aim, it uses first-order and second-order statistics assuming the Gaussian model on the data which makes the total number of parameters $(2D+1)K$ where D is the dimension of the feature vectors and K is the number of clusters in Gaussian Mixture Model. VLAD is similar to Fisher Vectors but it does not use the second-order statistics to reduce the number of parameters.

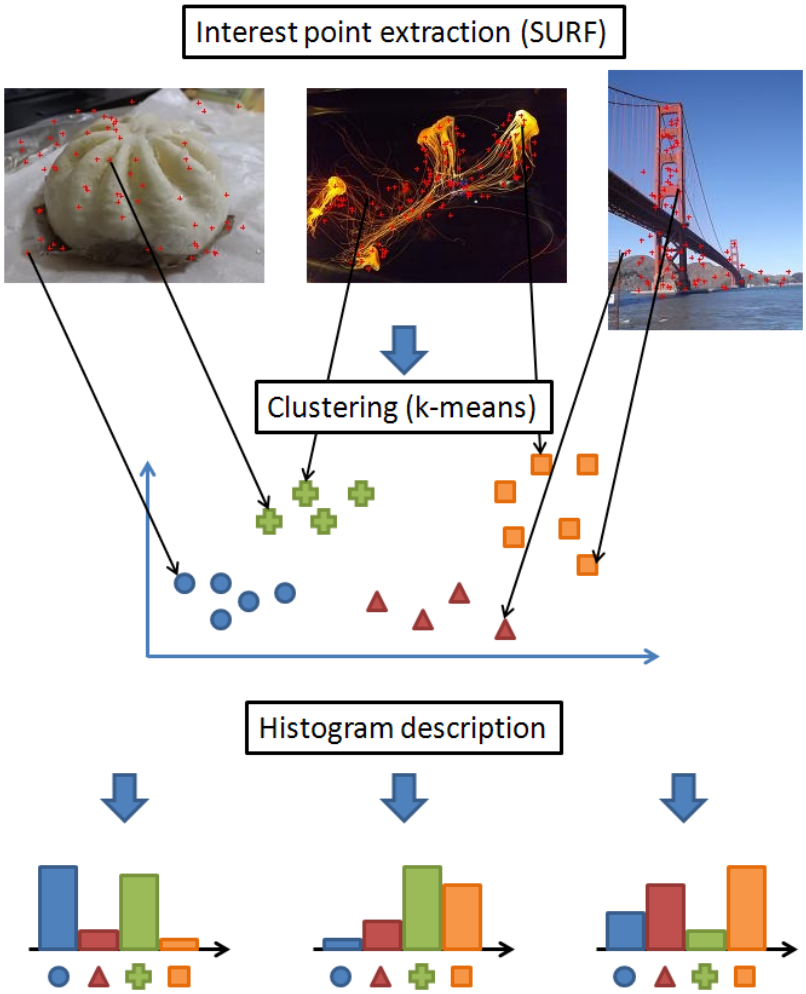


Figure 2.2: The general approach for hand-crafted based learning methods

From this point, crucial pre-deep learning AR studies will be introduced without entering into much detail.

2.1.1.1 3D Spatio-temporal Extension of 2D Spatial Detectors and Descriptors

Laptev extends the idea of the Harris corner detector to the spatio-temporal Harris corner detector [37]. Harris corner detector depends on the idea that the change in a window will cause sharp changes in both spatial directions around the corner. To do so, this technique uses first-order approximation and creates a matrix from the first-order derivatives and investigates the eigenvalues of this matrix for corner detection. In spatio-temporal Harris detector [37], the dimension is equal to three. Therefore, for a meaningful corner in the space-time domain, all of the eigenvalues should be high. For the scale selection, the maximum Laplacian of Gaussian (LoG) value is searched over the spatio-temporal domain. Next, from the selected scales, the corner points are re-calculated, which creates a two-step iterative algorithm.

In a different approach [10], the usage of 3D corner detectors is criticized due to the fact that in some behavior types, such as facial expressions, quite a few spatio-temporal corners are extracted. It is mentioned that sparseness is desirable to some extent but a rare number of interest points might not be sufficient for action recognition. The proposed interest points of [10] depend on the idea of perturbation in the temporal domain. For this aim, spatial Gaussian convolved with a quadrature pair of temporal 1D Gabor filters are applied to choose the interest points. Then, from the selected points, cuboids are defined which are spatio-temporal cubes from the video. From these cuboids, it is possible to extract various features like normalized pixel values, gradient information (G_x, G_y, G_t), and optic flow (V_x, V_y). Then, from these features, the BoW approach is followed for the representation and local histograms are created in order not to lose all space and time information. 3D-SIFT [54] is a study where the focus is more on creating 3D descriptors than the finding interest points. In this work, the interest points are randomly sampled from videos. The difference of 3D-SIFT representation from the 2D counterpart is that one more angle information also exists in histogram bins and sub-histograms are divided not only in the spatial domain but also in the temporal domain.

The authors of extended SURF [73] argue that the iterative 3D Harris-Laplace [37] detector is quite complex and might diverge since the iterative approach is applied for every feature detected. It is also argued that cuboids [10] are not scale-invariant,

since the size of a cuboid is a hyper-parameter. The proposed approach in [73] is that both the interest point and its scale are determined by the determinant of the Hessian matrix. Moreover, the second-order derivative calculation complexity is reduced by using the integral video concept. By using some box filters, the second-order derivatives are calculated and differently sized box filters are used to model the scale. Moreover, for the extraction of a descriptor, Haar wavelets are preferred and integral images are utilized for the calculation of these wavelet responses in order to decrease the complexity.

Another paper that utilizes the 3D Harris corner detector is [38]; however, for scale selection, a multi-scale approach is followed instead of an iterative algorithm to reduce the computational complexity. The cuboid concept is adopted and the scales of the cuboids are selected according to the scale of interest point. The grid size in cuboids is chosen as 3x3x2 and Histogram of Oriented Gradient (HOG) and Histogram of Oriented Flow (HOF) are used as features. Moreover, a spatial pyramid concept is followed to divide a video into spatial and temporal parts and calculating BoW separately for every part. For classification, non-linear SVMs are used.

The main focus in [35] is about creating a 3D descriptor which is a similar topic examined in 3D-SIFT [54]. The authors in [35] argue that the 3D-SIFT description proposed in [54] creates the problem of singularities at the poles because bins get smaller and smaller. To solve this problem, they propose making orientation with regular polyhedrons instead of parallels and meridians approach in the 3D-SIFT description. Moreover, the gradient calculation for different scale, they propose the integral gradient image approach, reducing the memory requirements. However, in my view, this paper does not propose rotation invariance as in 3D-SIFT or 2D-SIFT.

2.1.1.2 Trajectory-based Detectors and Descriptors

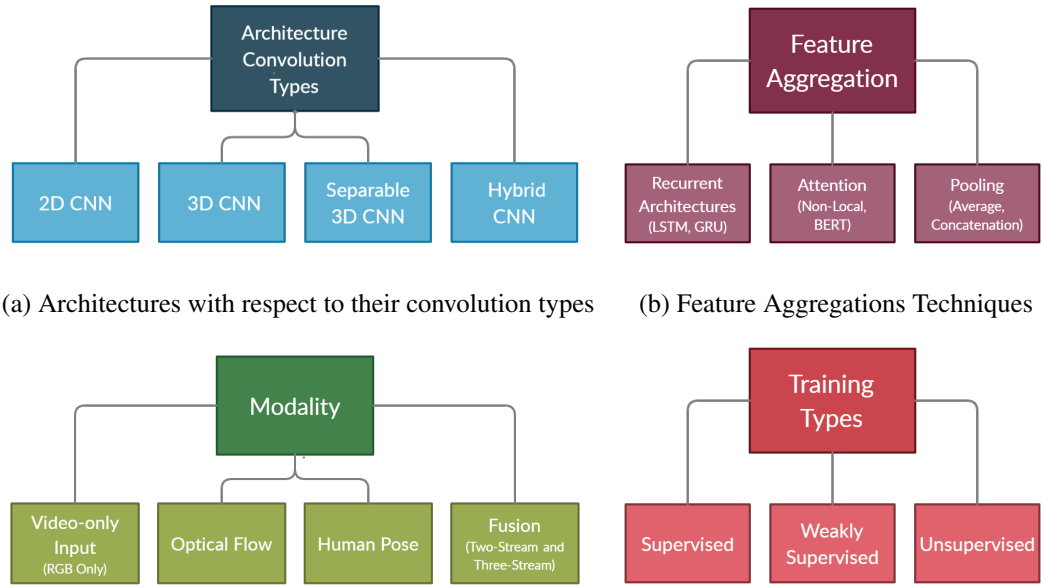
As mentioned before, another way of providing spatio-temporal characteristics to interest points is by creating trajectories from the detected spatial interest points. Instead of finding the interest points from the whole volume, *trajectory-based interest point detectors* firstly find spatial interest points in specific frames and secondly tracks them along the temporal dimension with specific temporal length. In order to create

trajectories, both KLT trackers [45, 46] and SIFT descriptor matching [29] are used.

Dense Trajectories (DT) [66] is a method that utilizes trajectory-based interest point detectors. Differently from the previous methods in the literature, it uses optic flow [15] for tracking instead of KLT tracking or SIFT matching. Another difference from the previous approaches is that it uses dense sampling for interest point selection instead of sparse selection. From the location of the detected interest points, the algorithm defines volumes and extracts three types of features from these volumes. These features are Histogram of Gradient (HOG), Histogram of Flow (HOF), and Motion Boundary Histograms (MBH). Among these features, MBH is found to be the best compared to HOG and HOF. The success behind MBH features is the fact that it suppresses camera motion. MBH is calculated by the first-order derivative of optical flow. Additionally, to these three types of features, the trajectories themselves are added to the created descriptions. In order to provide the DT algorithm with scale invariance, different trajectories are created from different scales.

Improved Dense Trajectories (IDT) [67] is the improved version of DT. In this implementation, the homography matrix between the consecutive frames is calculated. As a result of this estimation, camera motion components are removed from optical flow. Homography matrix calculation is performed with RANSAC on SURF descriptors. Moreover, instead of using the classical BoW method, Fisher vectors are used to encode the information of descriptors.

All aforementioned techniques have a common drawback: The spatio-temporal features are "hand-crafted" which means their design depends on human experience and intuition. There is no convincing reason to make all these representations yield optimal results for the AR problem. The solution to this fundamental problem has emerged in the last decade as learned representations that are obtained through deep neural networks. The next section will examine the deep-learning-based solutions specific to AR research.



(a) Architectures with respect to their convolution types

(b) Feature Aggregations Techniques

(c) Input Modalities

(d) Training Types

Figure 2.3: Taxonomy of deep learning methods for AR literature.

2.1.2 Deep Learning AR Literature

In this section, a general overview of deep learning methods is conveyed. For this aim, four different categorizations are considered as presented in Figure 2.3. These four categorizations are determined according to architecture convolution types, feature aggregation techniques, input modalities, and training types.

For the categorization according to the architecture convolution types, there are four approaches: 2D CNN [57, 17, 48, 69, 70], 3D CNN [62, 5, 8, 24, 16, 49, 6, 63], separable 3D CNN and Hybrid CNN[64, 76] in which the mixture of the others can be implemented.

For the categorization based on the input modalities, there are also four options: video-only input (RGB Only), extracted optical flow of the video, extracted human pose [12, 13, 78, 7] of video, and fusion in which the mixture of others can be performed as two-stream and three-stream architectures.

For the categorization in terms of feature aggregation, there are three methods: recurrent architectures [48, 11, 13, 40] (Such as LSTM, GRU, and convolutional version of them), attention architectures [72, 21, 20, 18, 51] (such as Transformers [65], BERT

[9] and non-local blocks), and pooling [31, 17, 22] (such as average, concatenation, minimum and maximum). The attention mechanism can also be performed within the implementation of recurrent architectures [55, 58, 13, 40].

For the categorization depending on the training types, there are three methods: supervised, weakly supervised, and unsupervised. The supervised approach is the most general one for the classification task and all samples in a dataset have a specific label. Weak supervision can simply be defined as noisy labeling. Unsupervised implies the complete absence of the labels.

From this point, the literature for the deep learning based methods is divided into general categories in order to emphasize the core idea of the studies for AR problem. These are 2D CNN architectures (Section 2.1.2.1), 3D CNN architectures (Section 2.1.2.2), recurrent architectures, pooling and fusion techniques (Section 2.1.2.3), attention models (Section 2.1.2.4), optical flow networks (Section 2.1.2.5), pose networks (Section 2.1.2.6) and weakly supervised and unsupervised techniques (Section 2.1.2.7). Note that these sections do not strictly follow the categorizations in Figure 2.3.

It is important to note that studies from the different categories might not be unrelated to each other. On the contrary, they can be strongly related to each other. For example, 2D CNN architectures might exist in all sub-categories. Pose and optical flow are different input modalities, therefore they cannot be considered independently from the architecture techniques. Additionally, most attention-based networks are utilized with the recurrent architectures. The main aim of this categorization is to convey the main focus of the studies and to relate similar studies to each other.

2.1.2.1 2D CNN Architectures

2D CNN architectures are trained basically with single images. However, it is possible to combine multiple frames in the channel dimension of 2D CNN architectures and this idea is implemented for the flow stream of two-stream networks. From that perspective, this is also in the category of optical flow networks which will be detailed in Section 2.1.2.5.

The breakthrough of deep learning methods for the action recognition problem starts with the introduction of two-stream networks [57]. The proposed architecture is not a very deep 2D CNN architecture and consists of only five convolutional layers and two fully-connected layers. The reason for the “two-stream" naming is the fact that there are two streams: one stream for RGB images as an input modality while the other one for optic flow images as an input modality. It is asserted that the RGB stream emphasizes the appearance of the action while the optical flow stream emphasizes the temporal information in the action. However, the temporal coverage is limited because only 11 frames (10 optic flow images) are used within the channel dimension of the first layer of the 2D CNN architecture.

The methods in [69] and [70] carry the two-stream concepts to deep 2D CNN architectures. For this aim, various good practices are proposed to train these deep 2D CNNs. One of the practices is cross-modality training which is training the optic flow stream with Image-Net pre-trained weights. Another proposed practice is the multi-scale crop concept as a data augmentation technique which is convenient for action classification. It is claimed that high dropout is also quite useful for training on an action dataset.

2.1.2.2 3D CNN Architectures

The first proposed 3D convolution architecture in the literature is C3D [62]. Before this approach, the temporal modeling with convolution was performed using an optical flow CNN in two-stream architectures or time-domain pooling architectures; however, they are restricted to 2D convolution and temporal information is put into the channel dimension. The difference of 3D convolution is that kernels are designed in 3D and channels and time information are represented as different dimensions. The C3D architecture has 8 convolutions, 5 max-pooling, and 2 fully connected parts.

One of the successful implementations of 3D convolution is Inflated 3D (I3D) [5], in which 3D convolution is modeled in a much deeper fashion compared to C3D. There are two important novelties in this method. One of them is the introduction of the Kinetics dataset which is larger than the previously commonly used two datasets, UCF101[59] and HMDB51[36]. This solves the problem of insufficient data for 3D

convolution to some extent. In addition, the designed architecture is the direct 3D counterpart of Inception V1 architecture, which enables to use of pre-trained Imagenet weights in 3D architecture after some manipulations. In I3D networks, 3D architectures are trained for both RGB and optical flow input modalities.

Another proposed technique that uses the idea of 3D convolution is ResNet3D [24] which is very similar to I3D. The only difference is that instead of using the 3D correspondence of inception architecture, it uses the 3D correspondence of ResNet architectures. Moreover, this method examines the fact that training 3D architectures requires more data as in the case of the Kinetics dataset; otherwise, it leads to overfitting. Moreover, it investigates the fact that the Kinetics dataset enables performance gains with deeper architectures as in the case of ImageNet. Different from I3D, this technique uses a 112x112 input size compared to the usage of 224x224 input size in I3D.

The downside of the 3D convolution architectures is their requirement for huge computational costs and memory demand. One of the solutions to model 3D convolution with reduced parameter size is creating pseudo-3D architecture, which is called Pseudo 3D network (P3D) [52] in which 3x3x3 spatio-temporal kernels are subdivided into 1x3x3 spatial kernel and 3x1x1 temporal kernels. These are also known as separable 3D convolution.

Another method that applies the separable 3D convolution concept is called R(2+1)D [64] and it is shown that dividing 3D spatio-temporal convolutions into spatial and temporal convolutions with the same parameter size has improved the performance significantly. In their paper, the performance of mixed architectures which consist of both 3D and 2D convolutions have also been evaluated. These mixed architectures replace certain 3D convolution layers with 2D and examine whether using 3D convolution in higher layers or lower layers is more beneficial. S3D [76] is the separable 3D convolution version of I3D. Similar to the study of [64], [76] also analyses the performance of mixed architectures which consist of both 3D and 2D convolutions. Additionally, [76] introduces the feature gating concept which seems to be an attention mechanism on channel dimension, which increases the performance of the architecture.

Another important 3D CNN architecture is Channel-Separated Convolutional Networks (CSN) [63], which depends mainly on the idea of depth-wise separable convolution proposed in [28], which is claimed to be a good trade-off between performance and efficiency. CSN proposes separating the channel interactions and spatio-temporal interactions, and CSN can be considered as the 3D CNN version of depth-wise separable convolution. CSN also investigates the group convolution which is similar to 3D ResNeXt architecture used in [24].

2.1.2.3 Recurrent Architectures, Pooling and Fusion Techniques

Pooling is a well-known technique to combine various temporal features; concatenation, averaging, maximum, minimum, ROI, feature aggregation techniques, and time-domain convolution are some of the possible pooling techniques [22, 48].

Fusion frequently used for AR is very similar to pooling. Fusion is sometimes preferred instead of pooling in order to emphasize pooling location in the architecture or to differentiate information from different modalities. Late fusion, early fusion and slow fusion models on 2D CNN architectures can be performed by combining temporal information along the channel dimension at various points in CNN architectures [31]. As a method, the two-stream fusion architecture in [17] creates spatio-temporal relationship with an extra 3D convolution layer inserted towards the end of the architecture and fuses information from RGB and optical flow streams.

Recurrent networks are also commonly used for temporal integration. LSTMs are utilized for temporal (sequential) modeling on 2D CNN features extracted from the frames of a video [48, 11]. E.g., VideoLSTM [40] performs this kind of temporal modeling by using convolutional LSTM with spatial attention. RSTAN [13] implements both temporal and spatial attention concepts on LSTM and the attention weights of RGB and optical flow streams are fused.

Moreover, Temporal Segment Networks (TSN) [70] is also an average pooling strategy with 2D CNN architectures. Instead of training the RGB stream with a single frame, TSN divides the video into segments, selects a frame from each segment and some segmental consensus are satisfied by averaging the final features of the selected

frames. With this implementation, the false labeling problem is reduced to some extent because the selected frame from a single segment might not contain characteristics of the action.

Slow-fast networks [16] can be considered as a joint implementation of both fusion techniques and 3D CNN architectures. There are two streams, namely fast and slow paths. The slow stream operates at a low frame rate and focuses on the spatial information similar to the RGB stream in traditional two-stream architectures, while the fast stream operates at a high frame rate and focuses on temporal information like the optical flow stream in traditional two-stream architectures. There is also some flow of information from the fast stream to the slow stream.

2.1.2.4 Attention

The spatial attention mechanism is firstly used in action recognition research by [55]. Spatial attention tries to direct methods to the related spatial parts of the action. Then spatially attended features are temporally modeled by LSTM.

Li [40] focuses on the idea of temporal modeling by using convolutional LSTM which is proposed in [56] for radar map forecasting. Besides, it adds spatial a attention mechanism to convolutional LSTM.

Another method that can be considered under the category of attention concept is non-local neural networks [72]. The authors aim to create a long-range relationship between the different spatio-temporal locations of features which cannot be obtained by the convolution operation. The method is inspired by non-local means which creates a relationship between distant pixels as a weighted sum of all pixels. It is claimed that transformers [65], which is an attention-based method, is the special case of their proposed algorithm. Their non-local blocks can be put into any CNN architectures.

Video action transformer network [20], where the transformer is utilized in order to aggregate contextual information from other people and objects in the surrounding video for the related bounding box of person. In [51], the multi-head self-attention mechanism of transformers is utilized as a self-attention mechanism in action recog-

nition for low-resolution videos. Actor transformers [18] utilizes transformers as an attention mechanism between the features of different actors. Differently from the video action transformer network, the actor transformer utilizes not only bounding box specific queries but also bounding box specific keys and values in the self-attention mechanism of transformers.

2.1.2.5 Optical Flow Networks

As mentioned before, the optical flow field (see Appendix A) is one of the important input modalities in two-stream networks. However, extracting optic flow needs pre-processing of the data and is more computationally complex compared to the RGB stream. Therefore, some studies focus on reducing the complexity of extraction of optical flow and enable end-to-end training of two-stream architecture including the extraction of optic flow.

Zhang et al. [80] use motion vectors instead of optical flow vectors and motion vectors are coarser and noisy compared to optical flow vectors. Motion vectors are normally used in video compression for its fast implementation. In their paper, a motion vector CNN is trained with knowledge distillation [27, 42] which aims to train a student network by a teacher network. Knowledge distillation aims to obtain with student network as good performance as with teacher network with much less parameter. Instead, in this network, the parameter sizes of two CNN are the same but the quality of input to the student network is worse compared to the teacher network. This implementation is claimed to be 20x faster compared to traditional two-stream networks.

Ng et al. (2018) [47] use 3D convolution to learn multi-frame optical flow extraction and 2D convolution to learn the optical flow of 2 frames jointly with the loss for action classification. It is shown that the guidance of action classification loss to optical flow extraction improves the performance compared to the single usage of optical flow loss for action recognition. The problematic side of this implementation is that the performance of the algorithm is upper bounded by the traditional optical flow algorithm which supervises CNN.

Wu et al. (2018) [74] try to use the information of P frames which is used in com-

pressed video formats, like MPEG-4 and H264. In video compression, P frames are referenced to the image frame with motion vectors and residuals. In this work, instead of referencing P frames to the previous frame, they are referenced to the latest I frame, which reduces the noisy output of motion vectors and residuals and denoted as accumulated motion vectors and accumulated residuals, respectively. It is claimed in the paper that with 3 input modalities (I-frames, accumulated motion vectors, and accumulated residuals), they achieved good performances with low complexity.

Zhu et al. (2017) [81] tries to calculate optical flow with an unsupervised setting. It claims that supervised training of optic flow is upper bounded by the performance of optic flow extractor for the loss function. In this work, optic flow extraction is assumed as an image reconstruction problem. Given an image pair, optical flow vectors are estimated and one of the images of the pair is tried to be constructed by utilizing both the other image and optical flow vectors. Additionally, the system is trained with smoothness loss which are the derivatives of optic flow vectors and the Structural Similarity Index Measure (SSIM) loss. Then, this network is trained with an end-to-end fashion with an action classification network.

TVNet [14] proposes a good trick to implement the TV-L1 optic flow algorithm with CNN layers, which reduces computation time (Complexity reduction increases with larger batch size due to parallelization). Moreover, due to the nature of the implementation with CNN layers, the parameters of convolutional filters which calculate the gradient of image and motion vectors and divergence of brightness difference can be made learnable.

Representation Flow Networks [50] considers the idea of calculating optical flow vectors from the intermediate features of CNN architectures and extends the TVNet study for this aim. Moreover, this effort introduces the flow of flow concept which provides a model with longer-term flow representation. The authors test various layer outputs to find better flow feature representations. Additionally, the idea of feature flow is transferred from the 2D convolution domain to the 3D convolution counterpart. It is claimed that the final network yields very good accuracy results with significantly less complexity compared to the rival methods of it.

Sun [61] tries to use the relationship between the features of different times, which

is similar to [50]. For achieving this, gradient information in the features (calculated with Sobel operator) and difference information between the features are concatenated and conveyed to the upcoming layer. However, it is observed that it does not yield as good performance as Representation Flow Networks [50]. This might be derived from the fact that it lacks the iterative process as in TV-L1 (or TVNet).

Another promising idea for the utilization of optical flow information is related to the distillation concept, namely Motion-Augmented RGB Stream (MARS) [8]. For the distillation concept, there are two architectures which are called as *teacher* and *student*. The main concept is guiding the student architecture with that of the teacher. The distillation concept is utilized in the literature to decrease the parameter size and time complexity, such that the student architecture is less complex architecture than the teacher architecture and student architecture is aimed to yield the same results with the teacher architecture. However, in MARS, the goal of distillation is different. The input modalities of teacher and student architectures are optical flow and RGB, respectively. The main purpose of the distillation is the obtainment of the similar features between the student and teacher architectures, one is fed by optical flow and the other is fed by RGB. As a consequence, the information of the optical flow architecture is utilized without the burden of extracting the optical flow of the RGB input, speeding up the architecture, significantly.

2.1.2.6 Pose Networks

Pose networks try to benefit from the pose of the people in a video. In the literature, it can be observed that pose network concepts are used jointly with the attention mechanism. One paper that uses this concept is Recurrent Pose-Attention Network (RPAN) [12]. In this paper, some joints are grouped into parts. According to parts and joints, different spatial attention parameters are defined. In addition, the loss function of spatial attention is guided by the pose map of the image.

Another paper, which uses pose attention, is Recurrent Spatial-Temporal Attention Network (RSTAN) [13]. However, the main focus of the paper is not on the pose attention concept as in the case of RPAN. In this paper, temporal and spatial attention concepts are implemented jointly and attention weights are fused between two stream

architectures which use both RGB and optic flow as input modalities.

PA3D [78] tries to use the outputs of one of the fast multi-person pose detector, called OpenPose [2] (see Appendix B). OpenPose tries to estimate Part Affinity Fields (PAF) which is the part information between the joints and joints themselves. PA3D uses the features of the CNN backbone, PAFs, and joints as different input modalities. For the final prediction, the scores of joints, PAFs, and features are fused.

The PoTion is another method that uses OpenPose [7] joint information. Instead of adding a temporal dimension for every joint, it encodes the temporal information by colorizing.

2.1.2.7 Unsupervised and Weakly Supervised Techniques

Weak supervision methods and unsupervised methods are one of the crucial parts of the action recognition, since labeling a huge amount of data is quite challenging, while extra data is crucial to obtain better performances.

Order Predicting Network (OPN) [39] is one of the methods that use an unsupervised method. In this work, video frames that contain the large optical flow fields are selected, which is called motion-aware frame selection. Then, the frames are shuffled and the network tries to predict the real order previous to shuffling. Another similar work called Video Jigsaw [1] tries not only to predict the temporal order but also to predict the spatio-temporal order such that three frames are selected and every frame is divided into 4 patches, which makes the total number of patches equal to 12. However, instead of trying to predict all $12!$ cases, it creates sub-samples by maximizing the Hamming distance between the samples, which reduces the memory and computation needs.

Additionally, instead of predicting order from per-frame based features, predicting order from the K-framed clip features is also possible, called as 3DRotNet [77]. It is claimed that frame chunks or clips contain more information than the group of frames and guides the ordering network better. To highlight it more clearly, 3DRotNet uses 3D CNNs to extract features, while OPN uses 2D CNNs.

Contrastive Bi-directional Transformer (CBT) [60] uses BERT with the Noise Contrastive Estimation (NCE) loss. In this paper, 3DRotNet is trained with the S3D network by using the same unsupervised method in 3DRotNet. Additionally, by utilizing the extracted 40 features from the 16-framed clips, longer context is learned, which makes the representations more temporally informative.

It is shown in the I3D method [5] that training UCF-101 and HMDB-51 from the fine-tuned model from Kinetics yields better performances compared to training them from scratch. The question that comes into mind is whether a larger dataset than Kinetics might be more beneficial for learning better representations for AR problems. However, there is no larger dataset for the time being to test this hypothesis. [19] considers testing this using weak supervision. Weak supervision can be thought of as noisy labeling. In this paper, four datasets from Instagram videos have been created using the hashtags. The maximum amount of video among these datasets within this study consist of 65M videos which is about 200 times larger than the Kinetics-400 dataset. In this paper, the effects of various factors on the performance are analyzed such as the dataset types, the number of labels, and the pre-training data format. As the convolutional architecture, R(2+1)D [64] has been chosen for the experiments.

2.2 Prominent Deep Leaning Based Methods in AR Literature

This chapter examines the following concepts in detail: Bi-directional Encoder Representations from Transformers (BERT), group convolution and depth-wise convolution, 3D convolution, separable 3D convolution, non-local neural networks, SlowFast networks, motion-augmented RGB stream networks (MARS), multi-fiber networks (MFNET), and temporal shift modules (TSM).

2.2.1 BERT [9]

Bi-directional Encoder Representations from Transformers (BERT) [9] is a bidirectional self-attention method, which has significant superiority over other attention-based methods for Natural Language Processing (NLP) tasks. The bidirectional property of this method provides BERT to fuse the context from both directions, instead of

relying upon only a single direction, as in former recurrent neural networks or other self-attention methods, such as Transformer [65].

The single head self-attention model of (BERT or Transformer) is in general formulated as:

$$\mathbf{y_i} = \frac{1}{N(\mathbf{x})} \sum_{\forall j} g(\mathbf{x_j}) f(\mathbf{x_i}, \mathbf{x_j}), \tag{2.1}$$

where $\mathbf{x}$ is the embedding vector that consists of extracted visual feature of the current frame and its positional encoding; i indicates the index of the target output temporal position and j defines all possible combinations; and $N(\mathbf{x})$ is the normalization term. The function $g(\cdot)$ is the linear projection inside self-attention mechanism of BERT, whereas the function $f(\cdot, \cdot)$ denotes the similarity between $\mathbf{x_i}$ and $\mathbf{x_j}$. The function $f(\cdot, \cdot)$ can be explicitly written as $f(\mathbf{x_i}, \mathbf{x_j}) = \text{softmax}(\theta(\mathbf{x_i})^T \phi(\mathbf{x_j}))$, where the functions $\theta(\cdot)$ and $\phi(\cdot)$ are also linear projections from the learned feature space. The outputs of $\theta(\cdot)$, $\phi(\cdot)$ and $g(\cdot)$ functions are known as key, queue and values.

Similar work for action recognition is implemented by using the technique entitled as non-local neural networks (NN) [72]. A non-local NN uses a similar concept in every bottleneck block of ResNet, except the last block by using 1x1x1 CNN filters in order to realize g, θ, and ϕ functions. However, BERT exploits matrices in order to realize the same functions, utilizes the multi-head attention concept in order to create multiple relations with self-attention, and utilizes the positional encoding concept in order to preserve the position of the words.

In Natural Language Processing (NLP) tasks, multi-head attention is used to learn multiple relations. For example, homophone words are projected into the same embedding. Therefore, there should be a need to learn multiple relations. Contrary to RNN-based methods, the positional information of the words is lost because of the summation term in 2.1. Therefore, positional encoding is applied in a way that position vectors (or embedding) are added to the feature vectors or word embedding. Positional embedding can be made learnable or fixed. In the original paper that proposes the Transformer concept, it is argued that making it learnable does not make any difference; however, in BERT paper [9], learnable positional encoding increases the performances.

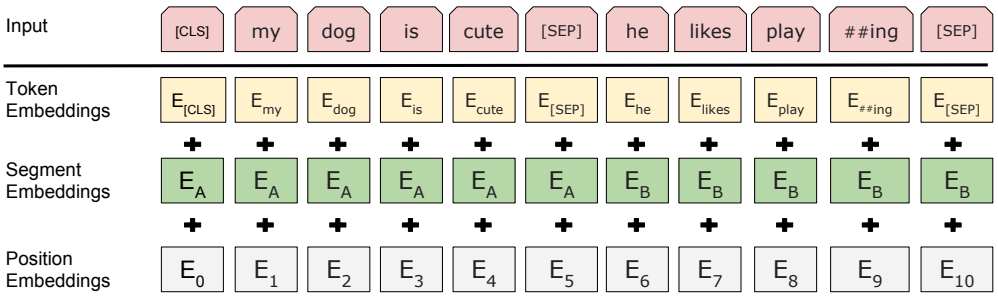


Figure 2.4: Masked LM in BERT [9]

The training of BERT consists of two stages. These stages are pre-training and fine-tuning. In the pre-train step, BERT is trained in an unsupervised manner. The pre-train stage is performed by a method called Masked LM (MLM). In this stage, some of the tokens of the words are replaced by a mask token. Then all of the words (masked or unmasked) are tried to be predicted by the BERT approach. Moreover, during the pre-train step, two sentences are given, which are either next to each other or irrelevant to each other and the proposed method is expected to predict that whether these are next to each other or not. For different sentences, segment (or sentence) embedding is added in order to provide the information about the relationship of words to sentences. For the prediction of the next sentence task, an extra classification token is added. From that position or index, the classification is performed. The MLM procedure in BERT is illustrated in 2.4.

2.2.2 Group Convolution and Depth-wise Convolution

Group convolution and Depth-wise convolution (DW) can be perceived as a way of making the architectures with fewer parameters and more computationally efficient. Group convolution and depth-wise convolution are related to each other. Depth-wise convolution is a special case of group convolution.

The illustration of group and depth-wise convolution is presented in Figure 2.5. The main concept for these convolutions is reducing the channel interactions for better efficiency. Only the channels within the group interact with each other. DW is a special case of group convolution where the number of groups is equal to the number of channels. In group convolution, the dimension of the filters changes from C_{in}xhxw

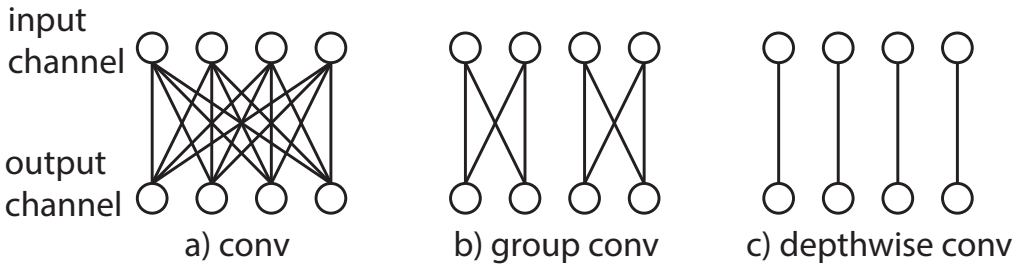


Figure 2.5: The demonstration for group and depth-wise convolution. Each circular node represents an input or output channel [63]. (a) A conventional convolution, the number of group is one. (b) Group convolution, where the number of groups is two in the figure. (c) Depth-wise convolution, the number of groups is equal to the number of channels and it is four in this example.

to $\frac{C_{in}}{K}$xhxw where C_{in} is the number of input channels, h and w are the horizontal and vertical dimensions of the filter, and K is the number of groups. In group convolution, both the number of parameters and computational complexity is reduced to one K-th of the conventional convolution. A better demonstration of the comparison between conventional and group convolution is shown in Figure 2.6.

However, it should be noted that reducing the channel interactions is not a good practice because the channels might contain complementary information and it might be processed jointly. Especially, in depth-wise convolution, the channel interactions are reduced to zero. Therefore, the depth-wise convolution is utilized within the concept of depth-wise separable convolution where the 1x1 channel convolutions are performed after 3x3 depth-wise convolution. Depth-wise separable convolutions are proposed in MobileNet architectures [28] as efficient CNN implementations. Similarly, the group convolution is applied only to 3x3 filters in bottleneck blocks of ResNet architecture and 1x1 convolutions are implemented conventionally (see Figure 2.10 for the bottleneck block of ResNet architecture).

2.2.3 3D Convolution

The contribution of 3D CNNs on the AR tasks is very crucial. For example, according to the I3D paper [5], the utilization of 3D CNNs has increased the performance

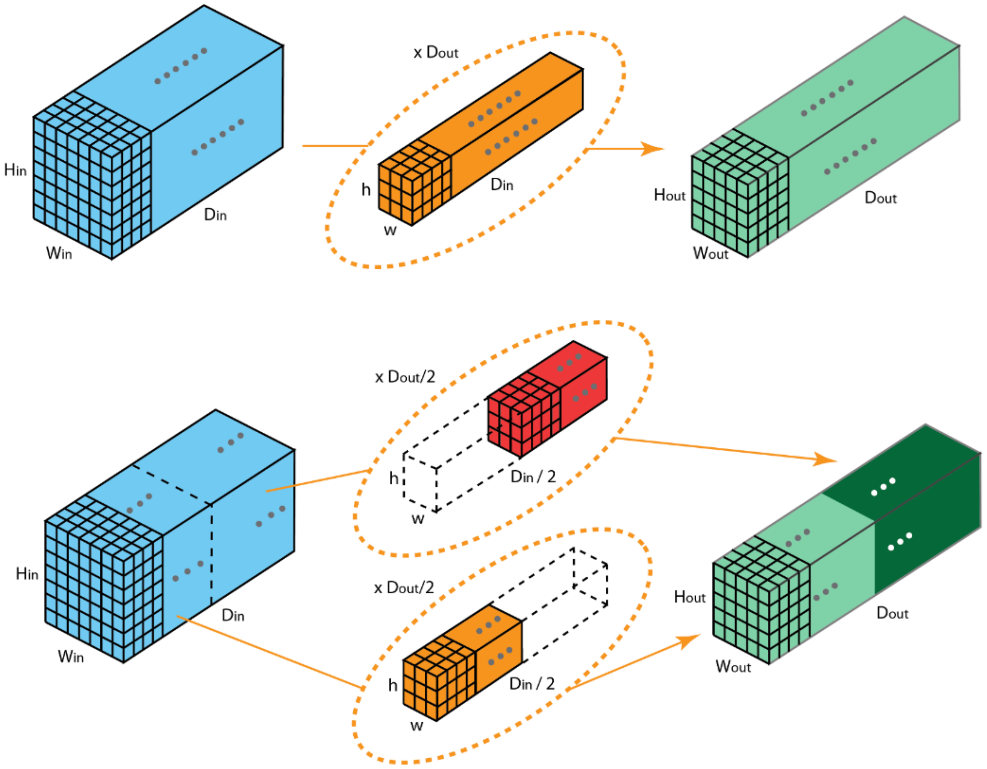


Figure 2.6: Conventional (Top) and group (Bottom) convolution operations. The image is from Towards data science: Comprehensive introduction to different types of convolutions in deep learning

with approximately 9% and 8% compared to 2D CNNs and LSTM-based architecture on the Kinetics dataset, respectively, all utilizing the Inception-V1 architecture. The reason behind the success of 3D CNN is the fact that the temporal information is utilized hierarchically and in an order similar to the spatial information. The pooling methods or recurrent neural network methods create temporal relations only at the final stage, which lacks successful temporal modeling. However, it should always be remembered that 3D convolution has quite a high number of parameters, which requires a huge training dataset in order to obtain successful performances. For example, in the same paper, it is reported that training from scratch and Kinetics pre-trained Two-Stream 3D CNN has obtained 66.4% and 81.2% performances on the HMDB51 dataset, respectively. This fact emphasizes quite clearly that the dataset is of great importance to benefit from any 3D CNN architecture.

Conceptually, the standard input training data format in 2D convolution consists of

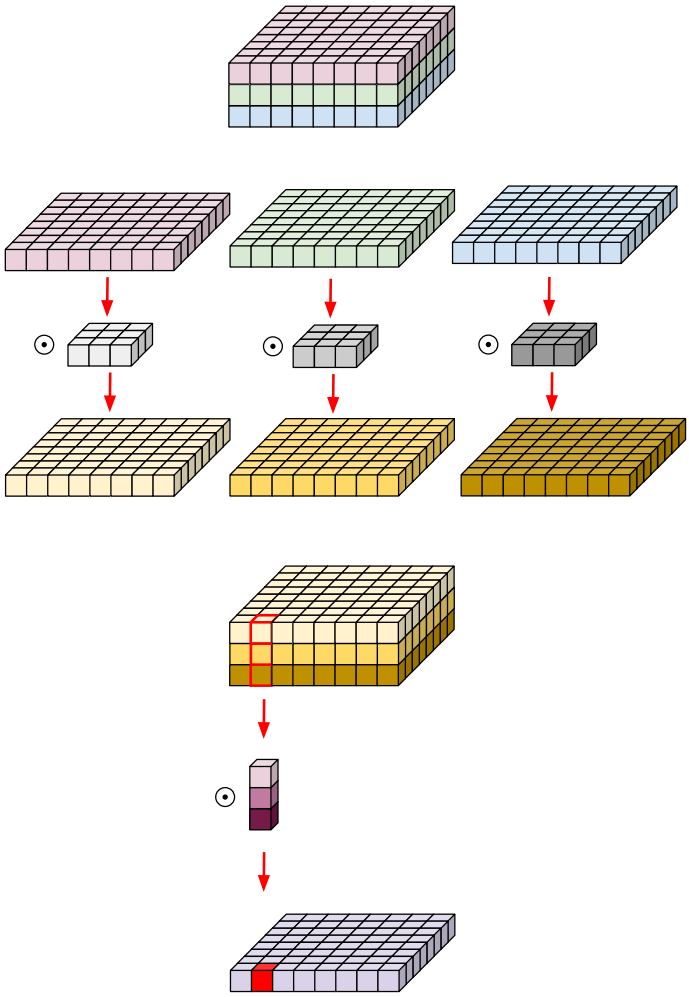
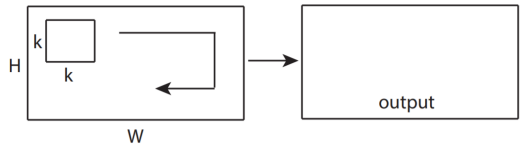
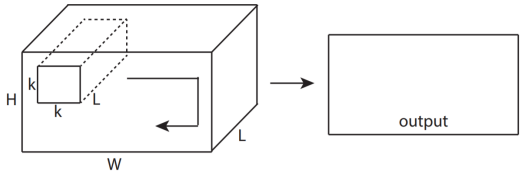


Figure 2.7: Depth-wise separable convolution. The image is from Eli Bendersky's website: Depthwise separable convolutions for machine learning

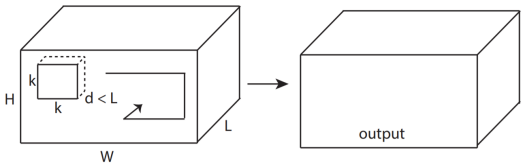
B, C_{in}, H, W, where B, C_{in}, H, W are to denote the batch size, the number of input channels, the dimension of height and width, respectively. The filter sizes can be defined as hxw where h and w are the filter height and width, respectively, and the real filter size is C_{in}xhxw if it is not group or depth-wise convolution. The output of *every* 2D convolution filter has a dimension of Bx1xHxW. In 3D convolution, the standard input data format is B, C_{in}, T, H, W, where T is added as a temporal dimension (i.e. video frames). The filter sizes can be defined as txhxw where t is the temporal length of the 3D filter, and the real filter size is C_{in}xtxhxw. The output of every 3D convolution filter has a dimension of Bx1xTxHxW. The different between 2D and 3D convolution can be seen in Figure 2.8.



(a) 2D Convolution



(b) 2D Convolution on multiple frames

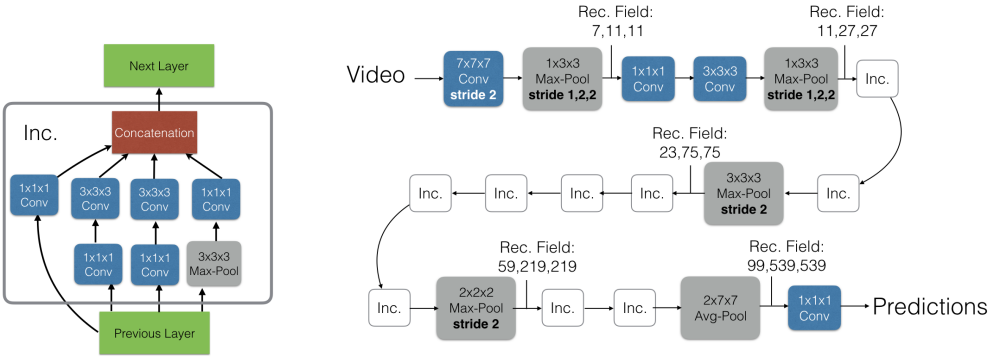


(c) 3D Convolution on multiple frames

Figure 2.8: 2D versus 3D convolution [62]

Mainly, there is two popular 3D convolution architecture category. There are Inception based and ResNet based since there are also popular 2D CNN architectures and using them enables the utilization of Image-Net pre-trained weights obtained in image classification which is a more mature area compared to the AR. To benefit from Image-Net, it is possible to consider pseudo-video which consists of the same frame repeating itself. Then, the weights in 2D CNN filters are repeated in temporal dimension with $\frac{1}{t}$ normalization factor where t is the temporal dimension of the 3D filter. As a consequence, the output of 2D CNN with the frame is the same with the output of 3D CNN with the pseudo-video which consists of the repeating same frame. In [5], it is also shown that starting with Image-Net yields better performance compared to starting from scratch.

However, it should also be denoted that the Kinetics dataset is sufficient and utilizing Image-net pre-trained weights as explained in the previous paragraph does not yield considerable performance improvements.



(a) 3D Inception module (b) 3D Inception-based CNN architecture: Inflated 3D (I3D)

Figure 2.9: Architectural information about 3D Inception network [5]

2.2.3.1 Inception Type Architectures

The inception-based methods consist of Inception blocks. There are two important aspects of this block. The first aspect is that it concatenates the outputs of differently sized filters, which creates diversity in the output. The second aspect is that before applying differently sized filters, it applies point-wise convolution, which reduces the size of the parameters of the network. The point-wise convolution is implemented with 1x1 sized filters. The 3D Inception module used in [5] is given in Figure 2.9a. The overall architecture of Inception-based 3D CNN architecture, namely Inflated 3D (I3D) can be examined in Figure 2.9b.

2.2.3.2 ResNet Type Architectures

The ResNet-based methods consist of basic or bottleneck blocks and these are two and three convolution operations, respectively [25]. A bottleneck block increases the number of filters with expansion size compared the basic block. These blocks gets the input, and apply a function $f(\cdot)$ and adds to itself, which can be shown in mathematical terminology as $x + f(x)$ or $g(x) + f(x)$, where $g(\cdot)$ is optionally applied where there is mismatch in channel sizes. Basic and Bottleneck blocks of 2D CNN architectures are shown in Figure 2.10.

ResNeXt [75] is a ResNet type architecture which includes a group convolution implementation (See Section 2.2.2 for more information about group convolution). The

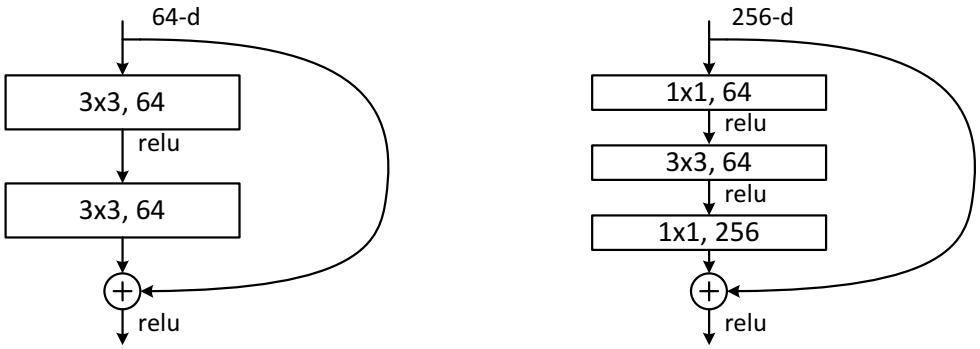


Figure 2.10: Basic (Left) and Bottleneck (Right) blocks of ResNet architecture [25]

group convolution filter sizes are $\frac{C_{in}}{K}$xhxw in 2D convolution or $\frac{C_{in}}{K}$xtxhxw in 3D convolution, where K is the number of groups and called cardinality of the architecture. It has been denoted that the ResNeXt-50 gets 1.7% higher results compared to ResNet-50 in Image-Net, which have nearly equal parameter sizes [75]. It is also shown that 3D ResNeXt is better than 3D ResNet architecture in activity classification tasks obtained on Kinetics-400 dataset [24].

Channel Separated Convolutional Networks (CSN) [63] is the depth-wise convolution version of 3D ResNet (see Section 2.2.2 for more information about depth-wise convolution). There are two proposed CSN architectures. The difference between the two architectures lies in the implementation of bottleneck blocks. These bottleneck blocks are called as Interaction-preserved channel-separated bottleneck block (ip-CSN) and Interaction-reduced channel-separated bottleneck block (ir-CSN). ip-CSN adds 1x1x1 convolution before the depth-wise convolution in order not to lose the relationship between the channels. The bottleneck blocks of ip-CSN and ir-CSN is visualized in Figure 2.11.

2.2.4 Separable 3D Convolution

Separable 3D convolution is the two-step implementation of classical 3D convolutions such that 3D convolution which has a filter size of C_{in}xtxhxw is replaced with the successive implementation of spatial filters which have size of C_{in}x1xhxw and temporal filters which have size of C_{in2}xtx1x1, respectively, and C_{in2} is equal to the number of filter of spatial filters before the temporal filters. The 3D convolution and

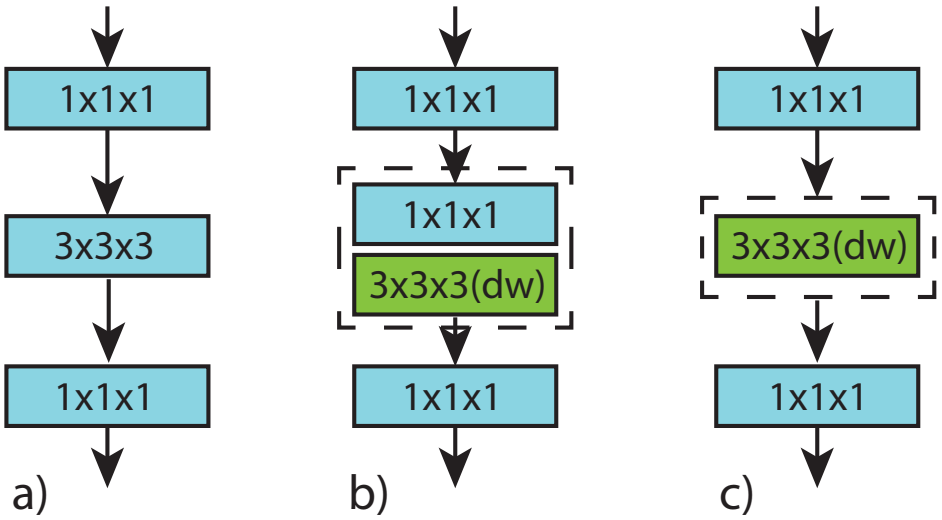


Figure 2.11: Standard and CSN bottleneck blocks. (a) Standard block. (b) ip-CSN block. (c) ir-CSN block [63]

separable 3D convolution is shown in Figure 2.12.

The pioneering work in this area is Pseudo-3D Nets (P3D) [52]. The traditional 3D bottleneck and proposed bottlenecks are shown in Figure 2.13. The performances are relatively close to each other, but the performance of P3D-A is a little bit higher, while it is a little bit less complex compared to others. Moreover, it is also possible to use all of them in the same architecture by following some order. It is claimed that using in the order of P3D-A, P3D-B, and P3D-C and again, increase the performance a little bit more compared to using only one type.

R(2+1)D [64] is another prominent work that follows the same idea with P3D. Additionally, R(2+1)D tries to analyze mixing 3D and 2D convolutions in the same architectures. It has been shown that R(2+1)D shows the best performance among all mixture variations of 3D and 2D CNN convolutions with the same number of layers. The bottleneck blocks utilized in the architecture are the same as the blocks of P3D-A. The 3D basic block of ResNet is also modified similarly. They argue that one of the positive aspects of separable 3D convolution is that it doubles the number of non-linear functions for the same number of parameters, resulting in learning more complex functions. Also, it is claimed that separable 3D convolution facilitates the optimization of the architecture.

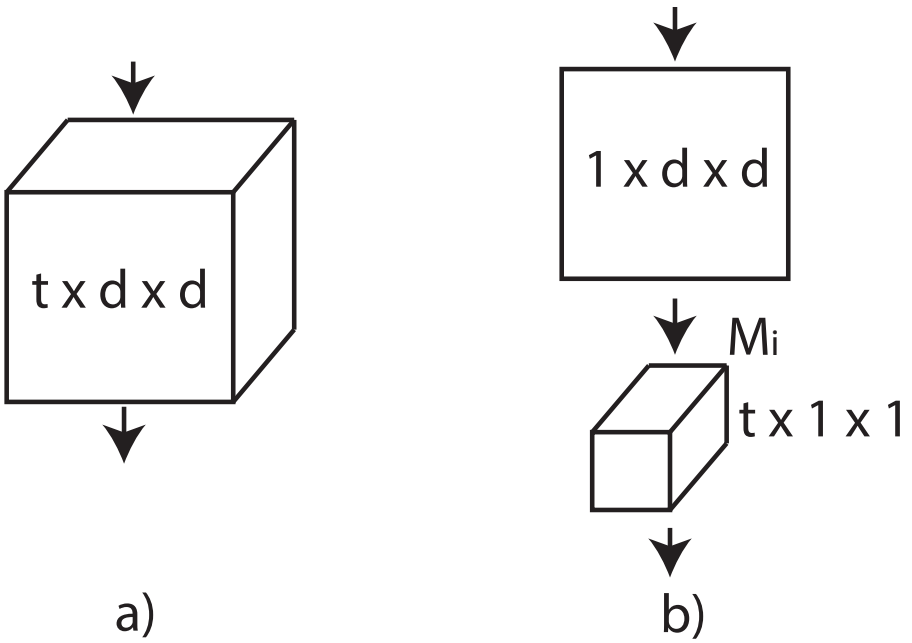
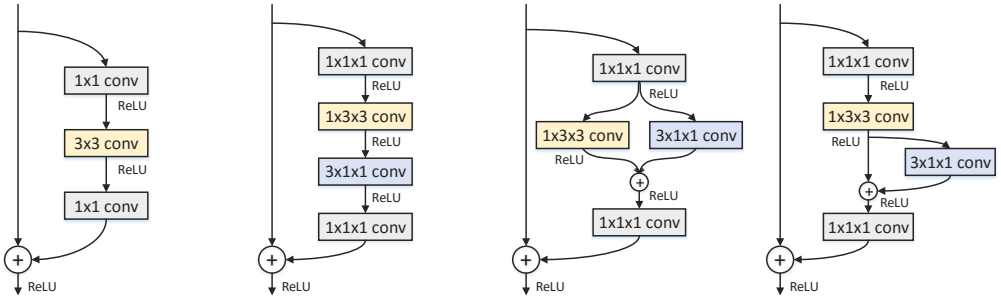


Figure 2.12: (a) Conventional 3D convolution. (b) Separable 3D Convolution [64]



(a) Residual Unit [7] (b) P3D-A (c) P3D-B (d) P3D-C

Figure 2.13: Proposed alternative separable 3D convolutional bottleneck blocks [52].

A quite similar idea is implemented in S3D architecture [76] which is the implementation of separable 3D convolution on I3D architecture.

In order to give more detail about the architecture of R(2+1)D [64], the classical spatio-temporal 3D convolution which has a filter size of 3x3x3 is converted into separable 3D convolution which consists of spatial convolution which has a filter size of 1x3x3 and temporal convolution which has a filter size of 3x1x1, respectively. In the utilization of this separable convolution, there is also a hyperparameter, namely the number of mid-plane channels, which is the number of channels between the spatial and temporal filters in separable 3D convolution. The number of mid-plane channels

in this architecture is determined in a way that the number of learnable parameters will be the same between the spatio-temporal 3D convolution and separable 3D convolution. The number of parameters in classical 3D convolution is IxOx3x3x3 and the number of parameters in separable 3D convolution is IxMx3x3 + Mx3xO, where I is the number of input channels, O is the number of output channels, and M is the number of mid-plane channels. Therefore, M is set to $\frac{I\text{x}O\text{x3x3x3}}{I\text{x3x3}+3\text{x}O}$ in order to equal the number of parameters between classical and separable 3D convolution. Besides, at the beginning of the R(2+1)D architecture, 45 2D filters of size 1x7x7 and 64 1D filters of size 3x1x1 are applied instead of utilizing 64 3D filters of size 3x7x7, both implementations have an equal number of parameters. However, it should be noted that some 3D ResNet architectures in the literature apply 64 3D filters of size 5x7x7 and 7x7x7 at the beginning of the architecture.

2.2.5 Non-local Neural Networks [72]

Convolution operations are applied within the local neighborhoods of the architectures. For capturing the long range dependencies, convolutions are applied hierarchically with stride and max pooling to capture long range dependencies layer-by-layer. A *non-local* operation concept is proposed [72] in order to capture long range dependencies directly within the layer. One possible advantage of using non-local is that the long-range relationship is created without losing the spatial context and detail as in CNN architectures. The non-local means concept is a traditional algorithm that relates all the pixels of an image to each other with some weighting. On the other hand, non-local neural networks aim the same on features instead of pixels and without a huge complexity. It should also be denoted that non-local concept is directly related with the attention concept.

The main idea behind the non-local concept can be explained by Equation 4.1. The general structure of a non-local block is showed in Figure 2.14. In the figure, T, H, W denote the dimension of spatio-temporal feature and are time, height, and width, respectively. 1024 in the figure is to denote the number of channels. This is given as an example, therefore 1024 may change depending on the layer on which a non-local block is implemented. With the help of $\theta(\cdot)$ and $\phi(\cdot)$ functions, the similarity is

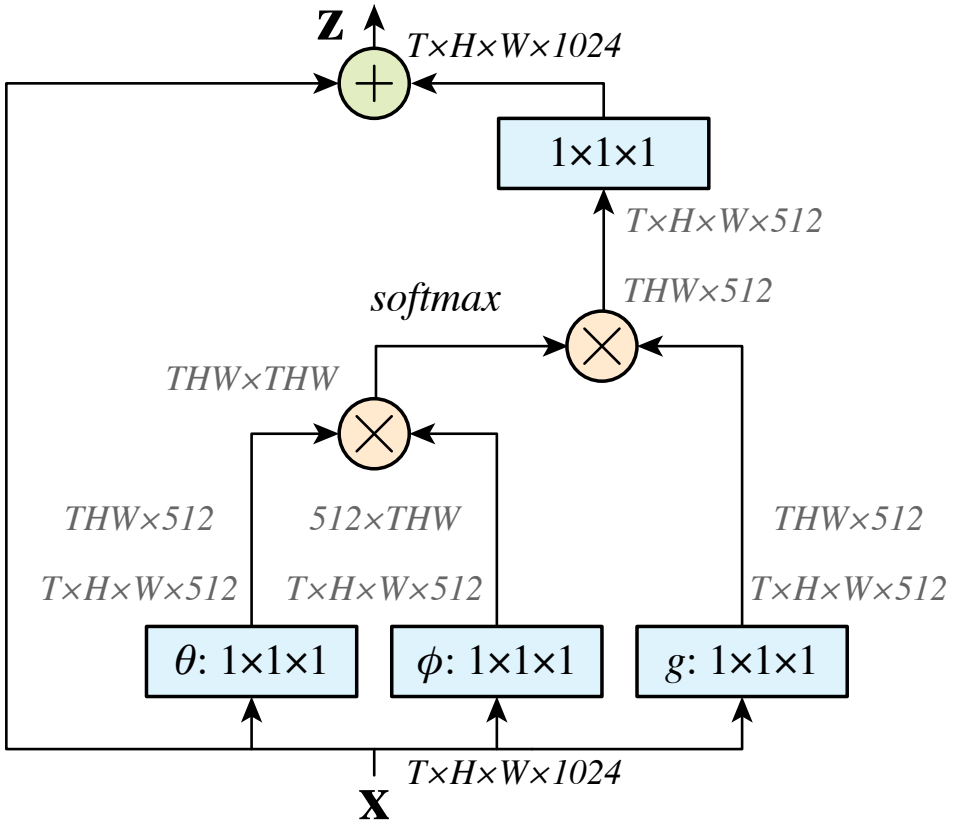


Figure 2.14: Non-local block [72]

calculated for every possible spatio-temporal positions of $\mathbf{X}$. These can be thought of as the relation of every position with the other all positions in spatio-temporal domain. Then, with the $g(\cdot)$ function, the features created across the channel dimension is moved to another domain that the aggregation of $g(\mathbf{X})$ accordingly with the calculated relation parameters yields good representations for every location in features space. The reason for the reduction of 1024 to 512 in the example of Figure 2.14 is due to the aim of the reduction of complexity of the architecture. It should be denoted that the output of a non-local block is added to the input. Therefore, this approach enables the utilization of pre-trained weights of any architecture.

When the architecture [72] is inspected, the authors implement non-local blocks in the second and third bottleneck groups (this is also called as layers in ResNet architectures). It should be noted that ResNet architectures have 4 bottleneck groups. For example, ResNet50 architectures consist of 3,4,6 and 3 bottleneck blocks for the bottleneck groups of 1,2,3 and 4, respectively. Moreover, in the bottleneck groups,

non-local blocks are not applied to all bottlenecks. Instead, odd numbers in the order are applied. For example, in the second bottleneck group which has 4 bottleneck blocks, the first and third one contains the non-local block.

The PyTorch implementation[1] of Non-Local ResNet50 architecture includes some modification with respect to the original ResNet architectures. Firstly, the bottleneck blocks which is originally implemented as 1x1x1, 3x3x3, and 1x1x1 is implemented as (3x1x1 or 1x1x1), 1x3x3, and 1x1x1, respectively (See Figure 2.15). The selection of bottleneck block types which are modified block-1 or modified block-2 (see Figure 2.15 for these types) is determined according to the block position in bottleneck group and these two types are used alternatively in specific group, one by one. As an exception to the first bottleneck group, all bottleneck blocks are modified block-2. Moreover, contrary to the traditional ResNet architectures [24], temporal stride is not implemented in the second, third, and fourth bottleneck groups. In this way, the aim can be preserving the temporal information throughout the CNN layers. The temporal size reduction is eight at the end of the architecture, which is 16 in ResNets of [24]. The filter size of the first convolutional layer is 5x7x7, which is 7x7x7 in ResNets of [24].

In this thesis, this explained architecture is denoted as "Modified ResNet50" and is analyzed in Chapter 3. Additionally, for $\phi(\cdot)$ and $g(\cdot)$ functions, max-pooling applied to the spatial domain (H and W) contrary the information given in Figure 2.14. The aim is to reduce the computational cost of the architecture.

2.2.6 SlowFast Networks [16]

SlowFast Networks [16] is a different perspective to the two-stream architectures. Instead of utilizing a two-stream or two-path structure for different modalities, SlowFast utilizes two-stream for a single modality and these two streams are not identical to each other (See Figure 2.16). As easily realizable from the architecture title, there are both slow and fast streams of pathways. The slow stream operates at a low frame rate and focuses on spatial information, such as RGB stream in traditional two-stream architectures, while the fast stream operates at a high frame rate and focuses on tem-

[1] github.com/Tushar-N/pytorch-resnet3d

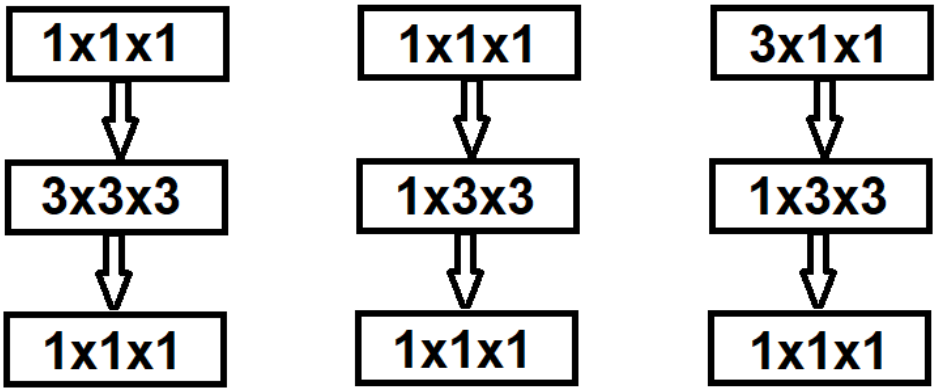


Figure 2.15: The modified blocks of the utilized ResNet architecture of non-local paper. Traditional block (Left), modified block-1 (Middle) and modified block-2 (Right)

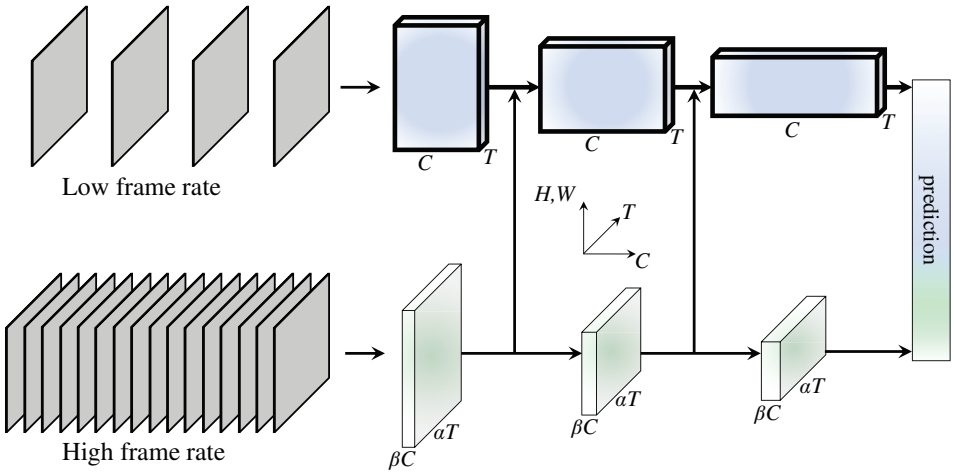


Figure 2.16: The Slowfast Network Architecture [16]

poral information as an optical flow stream in traditional two-stream architectures.

Conceptually, when a person waves or shakes a hand, the spatial information does not vary much; in other words, the hand is still mostly the same appearance and its properties (color, textural, brightness) evolve slowly in the time. On the other hand, the movement of the hand is also quite fast. Due to this fact, the fast stream does not focus on spatial information very much, the channel capacity can be less, therefore does not require temporal pooling in order to decrease the computational complexity. In other words, there is a trade-off between higher channel capacity and higher temporal resolution throughout the architecture, and preference is given to higher channel capacity in the slow pathway, and higher temporal resolution in the fast pathway.

It should also be noted that SlowFast architecture shares some similarities with the retinal ganglion cells in the primary visual cortex. The studies claim that about 80 % of these cells are Parvocellular (P-type) and 15-20 % are Magnocellular (M-type). P-cells provide fine spatial detail and color but have a slow response ability to stimuli. On the other hand, M-cells can respond in a faster manner but have a lower spatial ability. The analogy with the architecture is that the channel capacity is more on the slow pathway as the ratio of P-cells is much higher. As M-cell, the fast pathway has a faster response ability but has a low spatial ability, and as P-cells, the slow pathway has a slower response ability but has a better spatial ability.

Some of the hyperparameters of SlowFast architecture can be analyzed in Figure 2.16. As seen from the figure, the fast pathway has a lower channel capacity, such that β times of the channel capacity of the slow pathway. However, it has a higher temporal resolution, such that α times of the temporal resolution of the slow pathway. There are also lateral connections between the architecture which flows information from the fast stream to the slow stream. The typical α and β values utilized in SlowFast architectures are 4 or 8 for α, and $\frac{1}{8}$ for β.

The architecture utilized for the implementation of the SlowFast network is ResNet architecture. The details of an example SlowFast architecture with ResNet-50 is shown in Figure 2.17. As it can be observed in the figure, the slow stream is the same with the traditional ResNet architecture, such that the number of channels is set

stage	*Slow* pathway	*Fast* pathway	output sizes $T\times S^2$
raw clip	-	-	64×224^2
data layer	stride 16, 1^2	stride **2**, 1^2	*Slow* : 4×224^2 *Fast* : 32×224^2
conv_1	1×7^2, 64 stride 1, 2^2	5×7^2, 8 stride 1, 2^2	*Slow* : 4×112^2 *Fast* : 32×112^2
pool_1	1×3^2 max stride 1, 2^2	1×3^2 max stride 1, 2^2	*Slow* : 4×56^2 *Fast* : 32×56^2
res_2	$\begin{bmatrix}1\times1^2, 64\\1\times3^2, 64\\1\times1^2, 256\end{bmatrix}\times3$	$\begin{bmatrix}3\times1^2, 8\\1\times3^2, 8\\1\times1^2, 32\end{bmatrix}\times3$	*Slow* : 4×56^2 *Fast* : 32×56^2
res_3	$\begin{bmatrix}1\times1^2, 128\\1\times3^2, 128\\1\times1^2, 512\end{bmatrix}\times4$	$\begin{bmatrix}3\times1^2, 16\\1\times3^2, 16\\1\times1^2, 64\end{bmatrix}\times4$	*Slow* : 4×28^2 *Fast* : 32×28^2
res_4	$\begin{bmatrix}3\times1^2, 256\\1\times3^2, 256\\1\times1^2, 1024\end{bmatrix}\times6$	$\begin{bmatrix}3\times1^2, 32\\1\times3^2, 32\\1\times1^2, 128\end{bmatrix}\times6$	*Slow* : 4×14^2 *Fast* : 32×14^2
res_5	$\begin{bmatrix}3\times1^2, 512\\1\times3^2, 512\\1\times1^2, 2048\end{bmatrix}\times3$	$\begin{bmatrix}3\times1^2, 64\\1\times3^2, 64\\1\times1^2, 256\end{bmatrix}\times3$	*Slow* : 4×7^2 *Fast* : 32×7^2
global average pool, concate, fc			# classes

Figure 2.17: The details of the SlowFast-50 with $\alpha = 8$ and $\beta = \frac{1}{8}$ [16]

to 64, 128, 256, and 512 at the beginning of the bottleneck blocks for the first, second, third and fourth bottleneck groups in ResNet architecture, respectively and the bottleneck expansion is set to 4. However, for the fast stream, the number of channels is reduced β times, such that are these are 8, 16, 32, and 64 for $\beta = \frac{1}{8}$. Generally, for the SlowFast architectures, 32 frames with stride 2 are selected for the fast pathway, as in the implementation of non-local paper (See Section 2.2.5). The information about the lateral connections that fuse the information from the fast pathway to the slow pathway is not shown in Figure 2.17. The lateral connections are applied before every bottleneck group, namely res_2, res_3, res_4, and res_5 in Figure 2.17. In the lateral connections, the output of the fast stream is applied to another 3D convolution. For the 3D convolutions in lateral connections, there is also another parameter, called fusion convolutional channel ratio ($FCCR$). This parameter increases the channel capacity of the features of the fast stream before fusion. Therefore, the additional channel information fused to the slow pathway is the βx$FCCR$ times the number of

channels existing in the slow stream. Hence, the starting number of channels of first bottleneck blocks of the bottleneck groups (res_2, res_3, res_4, and res_5) are 80, 160, 320 and 640 instead of 64, 128, 256 and 512 for $\beta = \frac{1}{8}$ and $FCCR = 2$.

2.2.7 Motion-Augmented RGB Stream Networks (MARS) [8]

For AR tasks, the combination of the information from RGB and optical flow is quit beneficial in increasing the accuracy of the architectures. These architectures are known as two-stream architectures. However, extracting accurate optical flow vectors from the RGB input is an expensive process, resulting in a significant increase in the computational complexity of the architectures, limiting the utilization of two-stream architectures in real-life scenarios. During recent years, the researchers in this field put an effort to obtain the performance of the two-stream architectures without the need for the optical flow pre-processing.

Another brilliant idea for the aim of obtaining the performance of two-stream architectures without the additional complexity of optical flow calculation is Motion-Augmented RGB Stream Networks (MARS) [8]. The idea is related to the *distillation concept* in the related literature. Distillation denotes transferring the knowledge from a teacher network to a student network and it is first proposed by [27]. In this network, a student network is trained not only with the cross-entropy loss of the hard labels but also the cross-entropy loss with the soft labels of the teacher network which is pre-trained on the same task. In addition, some of the performance improvement brought by ensemble learning is partially obtained by the distillation of ensemble architecture to single architecture. Another form of distillation is the transfer of *privilege information* [42]. The utilization of privileged information depends on the idea that student and teacher architectures have different inputs but have the same purpose. By using this methodology, the regularization or another perspective from the teacher architecture to the problem might increase the performance of the student architecture.

In MARS, teacher architecture is the one that gets optical flow images, and student architecture is the one that gets RGB images. The aim in the distillation of MARS is to transfer the information from the flow architecture to the RGB architecture. In this setting, optical flow is the privilege information; however, it should be denoted

that the distillation is not implemented in the level of soft classification labels but in the level of the output features of the backbone. The final aim is obtaining the performance of flow architecture without the complexity of the extraction of optical flow inputs.

For distillation, there are also two types of implementation. The first one is Motion-Emulated RGB Stream (MERS) in which there is only MSE error between the extracted features of flow and RGB architectures. In MARS, additional classification loss from the RGB architecture is also backpropagated. The general loss function of the method can be defined as:

$$\text{Loss} = \text{CrossEntropy}(s_{RGB}, y) + \lambda||(\mathbf{f_{RGB}} - \mathbf{f_{flow}})||^2, \tag{2.2}$$

where $\mathbf{f_{RGB}}$ and $\mathbf{f_{flow}}$ are the final outputs of the RGB and flow architectures, $F(\cdot)$ and $G(\cdot)$, respectively, for a given input image $\mathbf{x}$. s_{RGB} is the predicted classification of the RGB architecture, which can be defined as:

$$s_{RGB} = \arg\max C(F(\mathbf{x})) = \arg\max C(\mathbf{f_{RGB}}), \tag{2.3}$$

where $C(.)$ is the classification function from the features. Larger α values drive the architecture to MERS, while smaller one makes it more MARS type. For the selection of the architecture, ResNeXt101 is chosen as the base architecture which is also utilized in [24] (See Section 2.2.3.2).

2.2.8 Multi-Fiber Networks for Video Recognition[6]

Multi-Fiber Networks (MFNET) [6] are proposed in order to reduce the number of parameters and the number of operations in the bottleneck blocks of the ResNet architectures. The idea of MFNET depends on the idea of group convolution which is used in ResNeXt [75] and Mobile Networks [28]. In Mobile Networks, the cardinality (group number) is equal to the number of input channels, namely depth-wise separable convolution. However, the authors of MFNET claims that despite the group convolution, the most of the architecture is not sliced (meaning not grouped) and dominates the computational cost.

In order to understand the advantage of group convolution better, consider the Figure 2.18a and Figure 2.18c which are basic block and multi-fiber block. Assume that

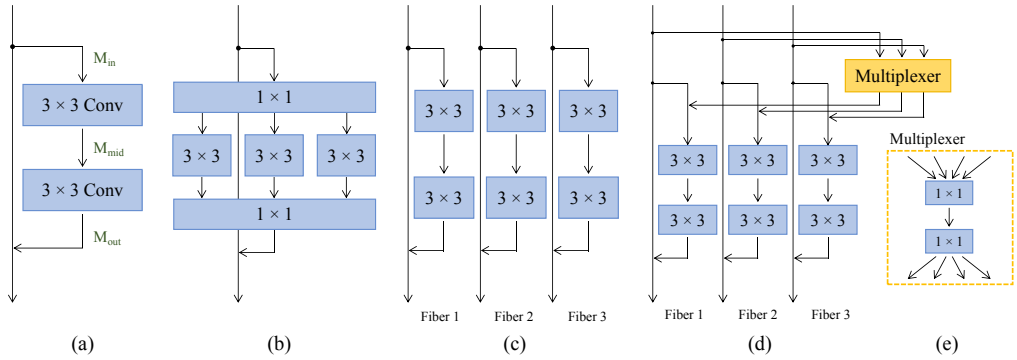
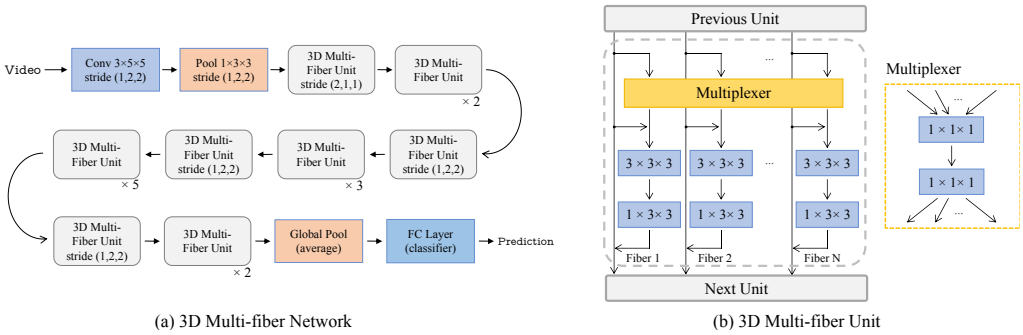


Figure 2.18: Possible bottleneck blocks implementations to ResNet architecture. (a) Basic Block. (b) Bottleneck block of ResNeXt atchitecture. (c) Multi-Fiber architecture. (d) Multi-fiber with multiplexer. (e) Multiplexer

the number of input channels to the first convolution is M_{in}, the number of output channels of the first convolution, and the number of input channels to the second convolution is M_{mid} and the number of output channels of the second convolution is M_{out}. The complexity of basic block can be written as $K\left(M_{in}\text{x}M_{mid} + M_{mid}\text{x}M_{out}\right)$ where K is constant related with the size of filter and the input tensor. However, when the group convolution concept is applied as in multi-fiber networks, the complexity is $KxN(\frac{M_{in}}{N}\text{x}\frac{M_{mid}}{N} + \frac{M_{mid}}{N}\text{x}\frac{M_{out}}{N})$ which indicates that group convolution has N times less complexity where N is the number of groups, cardinality or branch. However, the utilization of group convolution without creating a relationship between the channels possibly reduces the performance because preserving the complete channel interaction is also significant. With the aim of complete channel interaction, a new module namely multiplexer is added as in Figure 2.18d and this multiplexer module is demonstrated in 2.18e. The reason to use two convolutional operations instead of one is to reduce the complexity cost. When the number of channels between the two 1x1 convolutions is reduced by k, the computational gain is $\frac{k}{2}$.

For the 3D implementation of MFNET, original ResNet34 is modified in a way that the number of channels is changed. In the first convolution layer of MFNET, the preferred number of channels is reduced from 64 to 16 in order to reduce the computational complexity and the number of parameters because in the first layer the image resolution is quite high. Different from 3D ResNets implemented in [24], the temporal stride is implemented only one time at the beginning of the first bottleneck



(a) 3D Multi-fiber Network (b) 3D Multi-fiber Unit

Figure 2.19: (a) The architecture of 3D MFNET. (b) The bottleneck block or unit of the 3D MFNET architecture.

group. It should be remembered that the numbers of bottleneck blocks in the bottleneck groups are 2,3,5 and 2 for the first, second, third, and fourth bottleneck groups, respectively. Preserving the temporal dimension as much as possible might be important in order to create a better temporal relationship. The temporal size reduction is 2, which is 8 in modified ResNet50 (Section 2.2.5) and 16 in ResNet of [24]. The filter size of the first convolutional layer is 3x5x5. Generally, the traditional ResNet architectures have an output dimension of 2048, Inception type architectures have an output dimension of 1024 [5], while MFNET architecture has an output dimension of 768. The overall architecture and bottleneck block of MFNET architecture are shown in Figure 2.19.

A possible criticism of MFNET can be the preference of (224x224) input size dimension instead of using (112x112) dimension. For action recognition, utilization of the input size of (112x112) is frequently observed in popular architectures, such as [24, 64, 63, 19]. The selection of (112x112) over (224x224) as input dimension significantly reduces the time complexity of the architecture. One of the main aims of the MFNET architecture is stated as the reduction in the time complexity of the architecture. Therefore, the analysis for the input dimension of (112x112) should also be covered in the paper in my opinion.

2.2.9 TSM: Temporal Shift Module for Efficient Video Understanding [41]

Modeling the temporal information is one of the most crucial concepts in action recognition. In the literature, the temporal modeling is achieved mostly by two main

concepts: temporal modeling via recurrent architectures or various pooling strategies, such models are located mostly at the end of the architectures; hence, they are denoted as *late temporal modeling*. The other concept is the 3D CNN architectures which create temporal relationships in a more structured way throughout the architecture. The former one lacks creating a better temporal relationship compared to the latter one. However, 3D CNN architectures require high memory consumption and utilize lots of operations, resulting in higher time complexities. Therefore, a significant portion of the literature focuses on reducing the memory and computation demands, as indicated before.

Temporal Shift Module (TSM) [41] is introduced to the literature to obtain a higher accuracy at the level of 3D CNN architectures with the complexity of 2D CNN architectures. TSM modules can be plugged in any 2D CNN architectures and they do not increase the memory need and computational complexity but incurs data movement costs. The main idea behind the TSM is that some of the channel information belonging to the specific time index is transferred or shifted to the next or previous time index.

The visualization of the shift module is shown in Figure 2.20. In this figure, the features belonging to different time frames are shown in different colors. The tensor is shown along the temporal dimension in vertical, and along the channel dimension in horizontal. As shown in the middle image of Figure 2.20, the shift of some channels in both directions is a kind fusion technique between the features of different times. Shifting in both directions is an offline implementation. In a real-time application, in order to reduce the latency of the result, the instantaneous frame can be directly fed into the architecture. Therefore, the next frame is not known, and the shift is possible only in one direction.

The TSM module can be perceived in a way that the temporal information is processed in a structural way throughout the architecture as in 3D CNN architectures but with only the cost of 2D CNN architectures with an additional latency due to memory shifts. The penalty of the channel shift operation is calculated as a 13.7 % latency increase if all of the channels are shifted on a CPU inference. This latency increase is about 12 % in a P100 architecture.

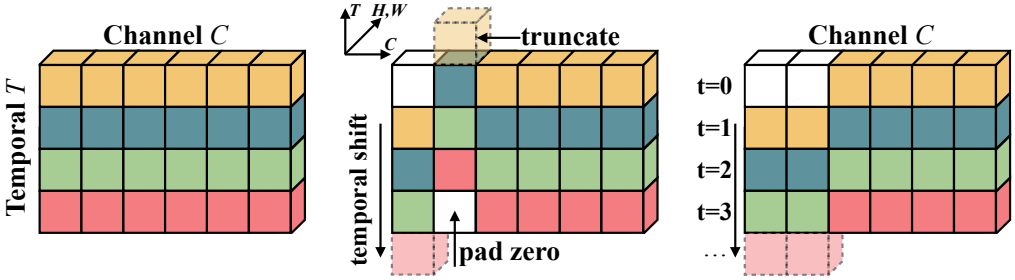
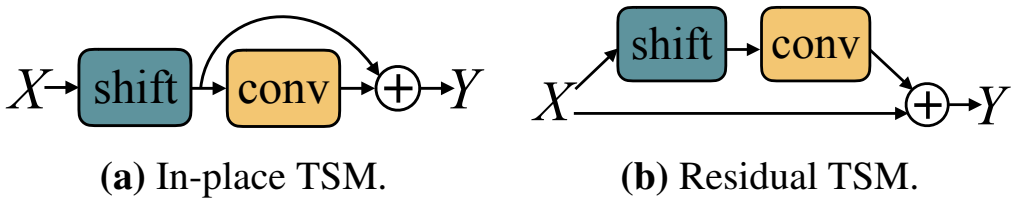


Figure 2.20: Left: The classical tensor structure. Middle: The bi-directional shift (Offline). Right: The uni-directional shift (Online) [41]



(a) In-place TSM. **(b)** Residual TSM.

Figure 2.21: Proposed TSM shift modules. (a) In-place TSM. (b) Residual TSM [41]

There are two important design considerations in the TSM module. One point is the ratio of the channels that are shifted. The other point is where to locate this module in a typical ResNet architecture. For the ratio of the channels that are shifted, increasing the information flow between different times increases the performance from one perspective. Nevertheless, from another perspective, increasing temporal information flow destroys the spatial information existing in the 2D CNN architectures. Therefore, there is a need to balance the ratio of the channel information flow. Another point is the location of the channel shift operation in the architecture. There are two types of proposed TSM modules which are In-place TSM and Residual TSM, which are illustrated in Figure 2.21. It is observed that Residual TSM yields better results compared to In-place TSM. One possible reason for this result can be explained as that the placement of the TSM module to the residual branch does not destroy the original feature format of the current frame.

The best result of TSM among the possible implementations on the Kinetics dataset is obtained with $\frac{1}{4}$ channel shift ratio with residual TSM. ResNet50 architecture is selected for the implementation of the TSM module. TSM is applied to every bottleneck block. The latency increase of $\frac{1}{4}$ channel shift ratio is between 3 % and 6 % for

CPU, TX2, and P100.

In order to understand the pros and cons of these algorithms from the literature, a separate chapter is devoted to analyze and compare their experimental results, next.

CHAPTER 3

EXPERIMENTAL EVALUATION OF LITERATURE

The main motivation of this chapter is to examine and understand the real virtues and possible drawbacks of the algorithms in AR literature through experimentation. In this chapter, initially, the datasets for AR research are introduced. Then, the implementation details of the experiments are presented. Then, various analyses are implemented as two separate chapters which are experiments of 2D CNN architectures and 3D CNN architectures. These analyses cover topics such as late temporal modeling, input modalities, architecture techniques.

3.1 Datasets for AR Research

For the action classification task, some common datasets are utilized for fairly comparing different algorithms. These are UCF-101 [59], HMDB-51 [36], Kinetics [32], Something - Something V1 and V2 (SMT) [23], and IG-Kinetics-65M [19]. UCF-101 and HMDB-51 are relatively older compared to Kinetics and SMT. IG-Kinetics-65M is different from the other datasets because it is collected in a weakly supervised manner.

3.1.1 HMDB-51 [36]

HMDB-51 [36] consists of 51 action classes. These classes are listed under five main categories. These are general facial actions (such as smiling, laughing, chewing), facial actions with object manipulations (such as eating, smoking, drinking), body

Table 3.1: Summary table for activity recognition datasets

Dataset Name	# Samples	# Categories
HMDB - 51	6766	51
UCF - 101	13320	101
Kinetics - 400	306,205	400
Kinetics - 600	495,547	600
Kinetics - 700	650,317	700
Something - Something V1	108,499	174
Something - Something V2	220,847	174
IG-Kinetics-65M (IG65M)	65,000,000	359

motion only (such as climbing, walking, push up), human-object interaction (such as throwing, riding a horse, brushing a hair), and human-human interaction (such as hugging, kissing, punching). There are 6766 clips in total from the 51 action categories, each containing a minimum of 101 clips. These clips are extracted from the 3312 videos. The videos are mostly collected from movies. For this task, the literature has reached about 82% top1 accuracy for the year 2020. For the evaluation, the dataset is split into three categories and the results are given as the average over three splits. For every split, the dataset is divided into two categories which are training and test. There is not a separate validation division.

3.1.2 UCF-101 [59]

UCF-101 [59] consists of 101 action classes These classes are divided into five main categories. These are human-object interaction (such as hammering, applying lipstick), body motion only (such as push-ups, baby crawling, swinging), human-human interaction (such as salsa spin, band marching), playing musical instruments (such as playing violin, playing guitar, playing flute), and sports such as (surfing, rafting, rowing). There are 13320 clips in total from 101 action categories. The videos are mostly collected from Youtube and clips are extracted from the 2500 videos which suggest that the variation in the dataset is limited. Similar to HMDB-51, the dataset consists of three splits, and results are given as the mean results of the three splits.

3.1.3 Kinetics [32]

Kinetics [32] is the most popular dataset as of 2020 which enables 3D convolution architectures which have many parameters, requiring lots of samples to train. Due to the large size of the Kinetics dataset, the performances obtained by using HMDB-51 and UCF-101 with Kinetics pre-training have also been increased significantly. The dataset has 3 versions as of 2020, which are Kinetics-400, Kinetics-600, and Kinetics-700 in which the numbers denote the number of classes in each dataset.

Considering that Kinetics-400 is the first version of the dataset, the specifications of it are explained in the beginning. In Kinetics-400, there are 306,205 clips which are extracted from 306,205 videos, suggesting that variation in the dataset is better compared to HMDB-51 and UCF-101. The clip length is about 10 seconds. The actions can be analyzed under three main categories which are singular person actions (such as laughing, robot dancing, drinking), person-person actions (such as hugging, shaking hands, kissing), and person-object actions (opening present, dribbling basketball, mowing lawn, playing violin, washing dishes). There are also some main categories in which sub-actions exist such that in the music category, playing various instruments exists as different action classes or in the dance category, various dance classes, such as salsa, swing, tango exists. The two-category division problem (test and validation sets are the same) which exists in UCF and HMDB no longer exists in Kinetics (see next section). For each action, there are 450-1150 clips. From these clips, 50 clips are for validation and 100 clips are for the test, regardless of the number of clips existing in the dataset.

Among all the classes, the most challenging ones seem to be the eating classes, such as eating doughnuts because the objects eaten are very small or partially eaten. Face-planting, slapping, sneezing, sniffing, drinking, drinking shots, drinking beers, shooting basketball are among the hardest samples according to the results obtained in the dataset paper. The most confusing activities are reported as between riding mule and riding or walking with horse, hockey stop and ice skating, swing dancing and salsa dancing, strumming guitar and playing guitar, shooting basketball and playing basketball, and go on.

It is also important to analyze what the action classification accuracy ratio is in order to understand whether motion or spatial information is important for the action. It is observed with the experiments that motion is crucial for the activities rock scissors paper, sword fighting, robot dancing, air drumming, exercising arm with the flow/RGB performances 5.3%, 3.1%, 3.1%, 2.8%, and 2.5%, respectively. Spatial information is significant for the activities making a cake, cooking sausages, sniffing, eating cake, making a sandwich with the flow/RGB performances 0.1%, 0.1%, 0.1%, 0%, and 0%, respectively.

The other versions of the dataset, Kinetics-600 [3] and Kinetics-700 [4] have 495,547 and 650,317 clips, respectively.

3.1.4 Something - Something [23]

Something-something is a dataset that can be used for action classification. There are 2 versions of this dataset as of 2020. Different from the other datasets, the videos are not collected from movies or Youtube and labeled by crowdsource-workers. Instead, the crowdsource-workers are required to act. Therefore, the noise in the dataset is decreased compared to others. The dataset consists of content, such as "opening [something]" or "holding [something] in front of [something]". Something can be any object related to the actions and increases variety, which probably increases the chance of learning instead of memorizing. The important concept about learning the action (or features related to the temporal domain) is the fact that the action should not be understandable by using high-level features or a single image. The main aim of the dataset is to make the models learn good temporal representations and emphasize the low-level features, instead of high-level features. Another important thing that the dataset places importance on is temporal resolutions. In order to learn the action precisely, there should not be an out of context features about the action. Therefore, clip length is kept small compared to other datasets.

The first version of the dataset contains 108,499 video clips, each of which has a duration of 2-6 seconds. The train, validation, and test set size ratio is equal to 8:1:1. The videos from the same crowd-worker occur only one part of the dataset (e.g. only in the train). There are 23,137 different objects for something keyword in the dataset.

For the creation of the dataset, 1133 crowd workers worked and there is a 127.32 crowdsource-workers per class on average. On average, there are 620 clips per class, ranging from 77 to 986 per class. The targeted classes are basic concepts instead of cultural things. The owners of the dataset inform us that the action classes aim to teach a one-year-old child as an analogy. In order to reduce the risk of cheating from the object type, different activities are implemented with the same object. Otherwise, the action can be classified directly from the object, which is an undesired fact to learn better temporally related low-level features. Additionally, some pretending actions have been added in order to learn complex relations. For example, there are opening and closing actions, but also pretend to open and close actions. There are also some confusing classes, such as putting something on to something and putting something next to something, which requires algorithms to understand the relative positions of the objects.

The second version of the dataset which is denoted as something-something-V2 consists of 220,847 clips. There are 30,408 unique objects, which was 23,137 in the first version of the dataset. The number of classes is the same with the first version which is equal to 174. For the new release, every clip is verified by five crowdsource workers that the clips contain the description assigned to them. The height of the resolution is also increased from 100 pixels to 240 pixels.

3.1.5 IG-Kinetics-65M [19]

IG-Kinetics-65M [19] is a dataset that is collected in a weakly supervised manner. This dataset belongs to the Facebook AI group. As of 2020 April, there is no released version available. However released caffe2 pre-trained models with this dataset which are R(2+1)D [64], ip-CSN, and ir-CSN [63] are available[1]. The Pytorch model of R(2+1)D trained with this dataset is also available[2]. The dataset is important to analyze how the performance changes with a significant increase in the number of samples. The dataset is collected from the videos of publicly available Instagram accounts.

[1] github.com/facebookresearch/VMZ
[2] github.com/moabitcoin/ig65m-pytorch

The weak supervision of this dataset indicates that there is no direct supervision in the collection of the dataset, such that there is no person who watches and labels the videos one by one. The weak supervision is satisfied in a way that the videos on Instagram are obtained with a related hashtag of a specific activity. For instance, consider an activity *"catching a fish"*. Then, possible hashtags are *"#catchingafish", "#catchfish", "#fishcatching"*.. The activities of IG-Kinetics-65M are selected from the activity labels of the Kinetics-400 dataset. However, for 41 labels, there is not a sufficient number of videos in collected Instagram videos, which is denoted as a minimum number of 50 videos per action, making the number of labels 359. Contrary to other datasets, the video duration changes up to 60 seconds, indicating that the temporal noise is very high in the dataset because in a longer video the activity might be performed anytime in the video. Contrary to IG65M, the duration of videos in the SMT dataset is very short in order to overcome the temporal noise problem. IG65M is long-tailed extremely from the number of samples perspective. In other words, there is a strong imbalance. In order to mitigate this effect, the square root sampling strategy is followed.

In addition to IG-Kinetics, IG-Noun, IG-Verb, and IG-Verb-Noun datasets are also created. For the IG-Noun dateset, 1428 hashtags are found which match the 1000 class of the ImageNet dataset. For the IG-Verb dataset, 438 Verbs from Kinetics and VerbNet are used. IG-Verb-Noun is created from the combination of 1428 noun hashtags and 428 Verbs, making 10653 labels in total. These are the pre-trained model datasets. The success of these pre-trained models of these datasets is dependent on the similarity between the labels of the fine-tuning dataset and the pre-training dataset.

For the temporal selection perspective, three options are considered. One of them is short videos, which consist of 1-5 seconds videos; another type is long videos, which consist of 55-60 seconds videos; and the last type is long-center videos, which are the center 4 seconds portions of long videos. Short videos are better for localization of activity, while long videos have better diversity. It is concluded that, for a fixed video budget, longer videos are better. For a fixed duration video budget, shorter videos are better.

Another analysis is related to the pre-training options. There are two settings in the

pre-training of the architecture in this analysis. The first option is pre-training with 2D architecture by using images and fine-tuning with the inflated 3D version of the same 2D architecture. The second option is pre-training directly with 3D architecture. It is concluded that the second option is better if enough number of videos are available. However, if the Kinetics dataset is utilized in the second option, it is slightly worse than the first option.

In the paper of IG-Kinetics [19], various analyses were made. For the effect of the number of labels of IG-Verb-Noun on the performance, if the architecture is fully trained, the increase is limited with 1% until 1300 labels and then saturates. However, if only the fully connected layer is trained, the accuracy increase is about 9%, then the accuracy saturates. For the effect of the number of labels of IG-Kinetics on the performance which has comparatively very few labels compared to IG-Verb-Noun, utilization of fewer labels drastically lower the performance even in fully trained architectures. For example, the performance of the utilization of 32 labels is about 6% less than the utilization of all 359 labels.

3.2 Implementation Details

In this section, implementation details of the experiments are given under six different subsections which are data augmentation, pre-trained weights, optimization, batch size selection, validation procedure and input modalities.

3.2.1 Data Augmentation

For data augmentation techniques, the good practices that are utilized in [69] are primarily followed.

During cropping, *random cropping* is not used; instead, multi-scale cropping is followed which is suggested by [69]. For multi-scale cropping, there are 13 possible crop positions, which are center, upper left, upper right, lower left, lower right, top center, left center, right center, bottom center, upper left quarter, upper right quarter, lower left quarter, and lower right quarter, which are all shown in Figure 3.1. The

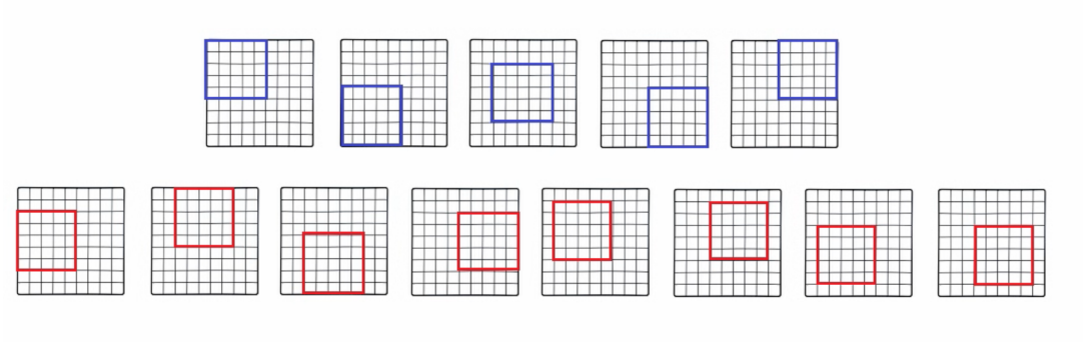


Figure 3.1: The crop positions for multi-scale cropping. Blues are used only inference.

horizontal flip is also applied with 0.5 probability. For scale augmentation, scales are determined as 1.0, 0.875, 0.75, 0.66. For both height and width, one of the scales from the pre-determined scales (1.0, 0.875, 0.75, 0.66) is selected randomly and if they are at most next to each other (maximum distortion concept), the selected scales are applied. For example, if the selected scales are (1, 0.875), it is suitable. However, if the preferred scales are (1,0, 0.75), which are not next to each other, the scale samples are re-selected. For most of the networks, the videos are resized to 256x340. Then after scale augmentation implementation, 224x244 input size should be given to the architecture by resizing the multi-scale cropped features. Similar to the suggestions of [69], a high dropout ratio is applied. The dropout ratio is determined as 0.8 for RGB-streams and 0.7 for flow-streams.

Additionally, similar multi-scale cropping techniques can also be followed during the inference of the architectures. For this aim, five crops which are center, upper right, upper left, lower left, and lower right are selected which are shown as blue boxes in Figure 3.1. With their additional horizontal crops, the extraction of ten clips is possible for every clip. Therefore, averaging the results of extracted 10 clips is possible implementation. This procedure during inference is called as *ten crop test* or *ten crop inference* in thesis.

3.2.2 Pre-trained Weights

For the training of 2D CNNs, the ImageNet pre-trained weights are used for RGB, optical-flow, and human pose input modalities. In order to utilize the ImageNet weight for optical flow inputs, the first layer filters are averaged through the channel dimension and replicated for the necessary input channel size. In other words, RGB input has 3 channel dimensions and filter sizes are 3xhxw. Then, the average across the channel dimension is calculated, which yields 1xhxw. Then, for the necessary input channel size, this average is replicated across channel dimension, such that C_{in}xhxw. In a traditional two-stream network, C_in is 20 because there 11 frames used in optical flow extraction, producing 10 frames optical flow output, and every frame has two flow channels which are x and y components of optical flow vectors. However, in this thesis work, generally, only one optical flow frame is used for temporal modeling in 2D CNNs, which results in two input channels.

For 3D CNNs, generally, the pre-trained weights of 3D CNN architectures trained on the Kinetics dataset are used. In the thesis work, the training on Kinetics is not implemented and Kinetics pre-trained weights are obtained from the related GitHub repositories.

3.2.3 Optimization

Optimization is a crucial concept to understand and utilize the true performance of any learning algorithm. A better algorithm with worse optimization might result in inferior performance compared to a worse algorithm with a better optimization strategy. In this comparison, ADAM [34], ADAMW [43], SWAT [33] and SGD optimizers are used. SGD has a constant learning rate in default but might have an adaptive learning rate with the first-order moment if used with the momentum concept. ADAM applies adaptive learning rates for different parameters with the first moment and additionally the second moment of the gradients. ADAMW adds weights regularization concept to ADAM and is claimed to have better generalization capability than ADAM. SWAT optimizer is the hybrid optimization of ADAM and SGD. At the initial stage, SWAT starts with ADAM optimizer but switches to SGD at the later stage of the optimiza-

Table 3.2: Optimizer result on RGB-ResNet-18-BERT architecture

	ADAM	SWAT	ADAMW	SGD
One Crop Validation	49.02	49.80	50.20	**50.49**
Ten Crop Test	51.50	51.11	**52.42**	50.78

tion.

It is interesting to note that despite the various recent optimizers, SGD has dominated the action recognition literature among various recent popular models [5, 76, 19, 16]. However, it is also known that BERT-based models generally prefer ADAM as the optimizer, such as [9], [60]. Therefore, there is a requirement to analyze the optimizer and determine one of them for the other training implementations.

The comparison of different optimizers is presented in Table 3.2. Based on this table, the ten crop test results show that the best result is due to the ADAMW optimizer. In center-crop validation, SGD is the best with a slight difference in performance compared to ADAMW. In ten-crop test analysis, the model which shows the best performance in one-center crop validation is chosen.

During all of the training steps, the learning rate schedule, namely *reducing learning rate on plateau methodology* is followed. The main purpose of this method is that when the desired loss or the desired precision does not change or does not show improvement, the learning rate is reduced to obtain better performance. In this method, there is a parameter, namely *patience*, which is how many epochs or iterations, learning rate does not change despite there is not any improvement. This parameter is set to 5 epochs, which means the learning rate is decreased to one-tenth if there is not an improvement for 5 epochs. For the best network selection in the validation, Top-1 precision is considered. For SGD, the learning rate is started from 10^{-2}, for the rest, it is started from 10^{-4}. The learning rate schedule is followed according to improvements of Top-1 at the beginning of this thesis but later changed to improvements of loss because the latter leads to improvements in some architectures such as I3D.

Due to all these results, all of the BERT models are trained with ADAMW, since ADAMW is faster than SGD and shows better performance. However, in traditional or standard architectures, SGD has been chosen unless stated otherwise in order to be consistent with the literature.

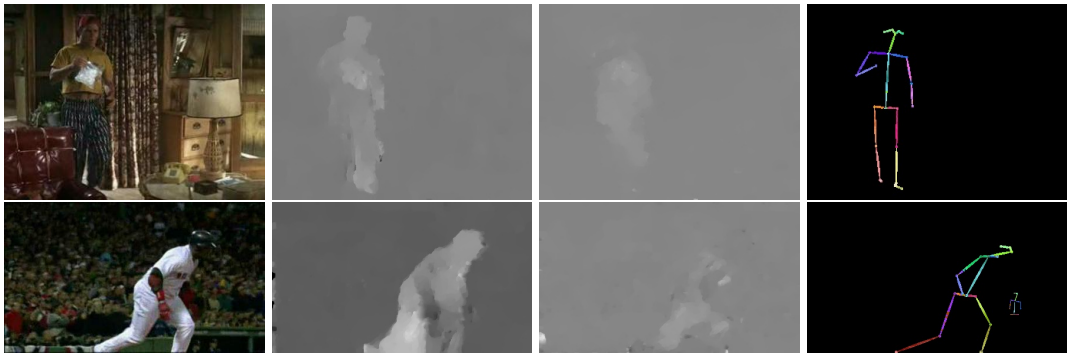
3.2.4 Batch Size Selection

The batch size of the architectures is selected as 128. However, the batch size of 128 requires significant memory utilization. Therefore, the iteration concept for weight update is utilized during the training of the architectures. For this concept, the maximum capacity of the batch size is loaded to the GPU or GPUs. Then, the number of iterations is set in order to obtain the effect of the batch size of 128. For example, if the maximum batch size is 8, the iteration size is set to 16 to obtain the same effect of the batch size of 128. One may argue that for the batch normalization, this might not show the same effect of direct usage of the batch size of 128. However, this is the most possible scenario with our resource-limited hardware.

3.2.5 Validation Procedure

For the validation of the architectures, frames or clips are selected from the center part of the segments. The segment concept is utilized for both of the training and validation of temporal modeling of the 2D CNN architectures. The detailed explanation of the frame selection procedure is given in Appendix-C Section C.1. The loss function of the architectures is the cross-entropy loss function.

For the selection of the best architecture, the epoch with the best Top-1 result is selected. This selection can be arguable since the calculated loss might be more convenient in order to prevent over-fitting. Therefore, the architecture selected with the best loss might perform better in multiple clip test or 10 crop test, since the validation is only performed to the fixed frames of the fixed clip of videos. However, the loss function is also arguable from the perspective the cross-entropy uses hard probabilities, ignoring the similarities between the classes. Therefore, a better option might be the selection of the best architecture with the mean of top1 and top3 results of the architecture. However, we implemented all the validation according to the best Top-1 result of the architectures.



(a) RGB (b) Flow X-component (c) Flow Y-component (d) Pose

Figure 3.2: Possible input modalities from the HMDB-51 walking and jumping classes

3.2.6 Input Modalities

The input modality concept mainly comes from the two-stream architectures. As mentioned before, optical flow is complementary information to the RGB input. In this thesis work, the input modalities are extracted by TV-L1 [79] algorithm (see Appendix-A). The motion vectors which have a magnitude value higher than 20-pixel displacement are clipped. Then, -20 and +20 are linearly mapped to 0 to 255 to represent the as an image. Additionally, the effects of the pose information are analyzed on action recognition by the extraction human poses as RGB images with OpenPose algorithm [2] (see Appendix-B). The visualization of input modalities is provided in Figure 3.2. X- and Y-components of the flow information are given together to the architecture as different channels of the input.

3.3 Experiments on 2D CNN Architectures

In this section, a detailed analysis is performed by the utilization of 2D ResNet type architectures. The first study of this part is the temporal modeling on the extracted features of 2D CNN architectures and the result of this study is presented in Section 3.3.1. The temporal modeling on the extracted features is also known as *late pooling strategy* in the literature. Secondly, the fusion of features from the different parts of the architecture is studied. Thirdly, the effect of the architecture depth (i.e. number of layers) on the performance is performed. Finally, the effect of the input modalities

(RGB, optical flow, human pose) on the performance is analyzed and two-stream and three-stream architectures, which are created by the fusion of these modalities, are inspected. In this section, the performances are reported as the average performance over the three splits of HMDB51, as it is preferred in most of the literature.

3.3.1 Late Temporal Modeling of 2D CNN Architectures

In this part, the following methods are implemented and compared with each other as late pooling strategies which is to create temporal relationship onto the extracted features of 2D CNN backbone:

- Convolutional GRU,
- LSTM,
- Average pooling,
- Concatenation pooling,
- Non-local attention,
- BERT.

In this analysis, the ResNet18 backbone is selected as a 2D CNN backbone with Image-Net pre-trainings. In this implementation, half of the Image-Net weights of the ResNet18 blocks are frozen because this is more memory efficient during the training and there is not a significant performance difference between the full update and half update of ResNet18 weights. The learning rate is set to $e-4$ for all architectures with ADAMW optimizer. The frame length is set to 16 for all of the *ResNet18 + Temporal Pool Types*, except *No Pooling* which is trained with a single RGB image. 16 frames are selected with equal intervals during the inference for all of them. The detailed explanation of the frame selection procedure is given in Section C.1.

For the settings of the selected late temporal modeling structures, the hidden size of convolutional GRU is set to 196. The number of inter-channels of non-local blocks (the dimension of attention mechanism) is set to 512 which is equal to the number of

Table 3.3: Results for temporal modeling on top of the 2D-RGB-ResNet18 on HMDB-51

Temporal Pool Type	**Top-1**	**Top-3**	**# Parameters**	**# Operations**
No Pooling	44.31	66.49	11.2 M	1.82 GFlops
Average Pooling	47.10	69.33	11.2 M	1.82 GFlops
Non-local + Average Pooling	47.56	69.76	12.25 M	1.87 GFlops
Concatenation	**51.04**	**72.20**	11.59 M	1.82 GFlops
Non-local + Concatenation	50.04	71.50	12.64 M	1.87 GFlops
LSTM	48.50	70.54	15.4 M	1.83 GFlops
Convolutional GRU	46.34	68.50	14.93 M	2.01 GFlops
BERT	49.17	71.31	14.36 M	1.83 GFlops

output channels of the ResNet18 backbone. The hidden size of LSTM is set to 512. BERT is implemented with eight attention heads.

The results of late temporal modeling on top of the RGB 2D- ResNet18 is presented in Table 3.3. In this table, Top-1, Top-3 results, the number of parameters, and the number of operations of *ResNet18 + Temporal Pool Types* are specified. Firstly, there is a necessity of highlighting the *No Pooling* which is denoted as one of the temporal pool types in Table 3.3. ResNet18 is trained with a single RGB image in *No Pooling* while the rest of them is trained with 16 RGB images. However, the inference of *No Pooling* is implemented similarly with *average pooling* type with 16 RGB frames. This is to highlight the importance of the implementation of TSN [71]. The implementation of TSN (average pooling over no pooling) increases the Top-1 performance with about %3. The only temporal pool type which worse than average pooling is convolutional GRU. The addition of non-local increases the performance of average pooling but worsens the performance of concatenation pooling. The best temporal pool type of the Table 3.3 is concatenation. This is the only analysis in this thesis that BERT is not the best. We consider that the best late temporal modeling might be dependent on both the CNN backbone and the modality. It is also possible that the optimization of the training of the BERT temporal pooling might not be performed well because of the sub-optimum hyperparameter for the specific architecture selection such as the learning rate.

Table 3.4: The Results of fusion types with 2D-RGB-ResNet18 concatenation pooling on HMDB51

Fusion Type	**Top-1**
No fusion	51.04
Dual Fusion	51.87
Triple Fusion	**52.02**
Quadruplet Fusion	51.13

3.3.2 Feature Fusion from the Different Parts of 2D CNN Architectures

In this part, the feature fusion from the different parts of the architecture is studied. Different parts of the architecture imply that instead of using features only from the end of the architecture, some features are extracted from the intermediate parts of the architecture as well. In order to convey where the features are created in the ResNet18 architecture, the ResNet18 architecture is presented in Figure 3.3. There are four block groups not only in ResNet18 but also in all ResNet architecture, which are denoted as conv2_x, conv3_x, conv4_x, and conv5_x in Figure 3.3. The selected baseline for this study is ResNet18 with concatenation pooling which is shown to the best in Section 3.3.1. The proposed fusion types are *dual fusion* (conv4_x and conv5_x), *triple fusion* (conv3_x, conv4_x and conv5_x), and *quadruplet fusion* (conv2_x, conv3_x, conv4_x and conv5_x).

The features at the end of the architecture might not contain all the necessary information that is created throughout the architecture, although these final features are created hierarchically from the features of earlier layers. For instance, the features of early layers of the architecture have a better spatial resolution. Therefore, fusion might be a good strategy in order to complement the final features of the architecture

The results of fusion study is presented in Table 3.4. Based on this table, it can be observed that dual fusion improves the no fusion strategy and triple fusion improves the dual fusion strategy. However, the addition of the features of the first block group to the other three features groups even makes worse the architecture than dual fusion. One possible reason might be the fact that to utilize the earliest blocks, the weight update is also enabled to the first block group (conv2_x in Figure 3.3), which is disabled in other fusion strategies. Therefore, preserving the Image-Net weights for the

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

Figure 3.3: Different Layers and their corresponding Output sizes for ResNet18 Architecture

first block group might be crucial.

3.3.3 Effect of Network Depth and Input Modality on 2D CNN Architectures

In this section, the effect of the architecture depth and variations in input modality on the performance of 2D CNN architectures are studied for action recognition. Firstly, for RGB input modality, the architecture depth analysis is presented. Next, the late temporal modeling is studied within the context of optical flow and human pose modalities. Then, the impact of different input modalities (RGB, optical flow, human pose) and two-stream (RGB+optical flow) and three-stream architectures(RGB+optical flow+human pose) is finally examined. For further information about the input modalities, see Section 3.2.6.

Effect of Network Depth: For this part, ResNet18, ResNet34, ResNet50 and ResNet101 architectures are selected among all ResNet-based networks. The selected temporal

Table 3.5: The effect of architecture depth on the performance of 2D-RGB ResNet with concatenation pooling and triple fusion on HMDB51

Architecture	**Top-1**	**# Parameters**	**# Operations**
ResNet18	52.02	# 11.91 M	# 1.82 GFlops
ResNet34	53.34	# 22.02 M	# 3.68 GFlops
ResNet50	55.44	# 26.21 M	# 4.12 GFlops
ResNet101	58.80	# 45.20 M	# 7.85 GFlops

modeling for this part is triple fusion concatenation temporal modeling. The analysis are implemented on three splits of HMDB51 with RGB input modality. The results of this analysis is presented in Table 3.5. As the architecture depth increases, the top1 performance also increases but with the cost of memory utilization and computational complexity,

Late Temporal Modeling on Optical Flow and Human Pose: For the late temporal modeling of optical flow and human pose modalities, average pooling, concatenation, concatenation with triple fusion and BERT are selected as possible candidates from Sections 3.3.1 and 3.3.2. The reason for selecting average pooling is its efficient implementation and its common utilization, and the reason for selecting concatenation, concatenation with triple fusion, ,and BERT is their success on RGB input modalities presented in Sections 3.3.1 and 3.3.2.

The result of this analysis is given in Table 3.6. Firstly, the best late temporal modeling strategy is BERT for both modalities. It should be noted that BERT is worse than the concatenation for RGB modality according to Table 3.3. Additionally, concatenation is even worse than the average pooling for optical flow modality. Another conflicting fact about the optical flow modality with the result of RGB modality is that triple fusion worsens the performance in concatenation pooling strategy. For pose modality, concatenation is a better temporal modeling than average pooling and triple fusion improves the performance of "no fusion", which is coherent with the result obtained on RGB modality.

Comparison of Modalities: For this part, the performances of different modalities are compared with each other. Additionally, two-stream (RGB+Flow) and three-stream (RGB+Flow+Pose) architectures are also analyzed. Two-stream and three-stream are late fusion strategies between the different modalities. Each stream is trained with

Table 3.6: Top-1 performances of late temporal modeling on ResNet18 backbone with optical flow and human pose modalities on HMDB51

Architecture	**Flow - Top-1**	**Pose - Top-1**
Average Pooling	51.68	44.14
Concatenation - No Fusion	48.58	44.75
Concatenation - Triple Fusion	47.03	45.82
BERT	**54.73**	**46.45**

different modality independently, and the normalized class scores of modalities are averaged during the inference. For this analysis, the temporal modeling type of the architectures is selected specifically to modality. Concatenation pooling is the best for RGB, therefore it is selected for this modality. BERT is the best for optical flow and human pose, therefore it is selected for these modalities.

The performances of different modalities, two-stream, and three-stream architectures are presented in Table 3.7. For this analysis two backbones are preferred which are ResNet18 and ResNet101.

Firstly, The performance increase of RGB modality with the depth of architecture is more dramatic than optical flow and pose modalities. This result is probably due to the fact that the pre-trained ImageNet weights are also trained with RGB modality, resulting in better utilization of RGB modality for the action recognition task.

Secondly, the results indicate that different modalities produce complementary information for each other. Two-stream architectures are quite popular in action recognition literature due to their benefits for increasing the performances. In this table, two-stream architectures (RGB+Flow) increase the maximum Top-1 performance of single streams about 8% and 6% for ResNet18 and ResNet101, respectively. Moreover, the addition of pose information to two-streams also increases about 3% and 4% of ResNet18 and ResNet101 backbones in action recognition, although single utilization of pose-stream is lower than RGB and optical flow modalities.

Table 3.7: Comparing modalities on HMDB51 using Top-1 for AR

Modalities	**ResNet18**	**ResNet101**
RGB Concatenation	52.02	58.80
Flow BERT	54.73	57.58
Pose BERT	46.45	49.59
Two-Stream (RGB concatenation + Flow BERT)	62.81	65.21
Three-Stream (RGB concatenation + Flow BERT + Pose BERT)	**65.56**	**69.15**

Table 3.8: Ablation Study on 3D-RGB-ResNet type architectures on Split-1 of HMDB51

Architecture	**Top-1**	**# Parameters**	**# Operations**
ResNet101-16f	61.18	85.35 M	13.95 GFlops
ResNeXt101-16f	63.14	47.63 M	**9.64 GFlops**
ResNet101-64f (16f pre-trained)	66.67	85.35 M	55.80 GFlops
ResNeXt101-64f (16f pre-trained)	68.56	47.63 M	38.56 GFlops
ResNeXt101-64f	73.07	47.63 M	38.56 GFlops
ResNeXt101-64f-224 (64f-112 pre-trained)	**75.23**	**47.63 M**	153.36 GFlops

3.4 Experiments on 3D CNN Architectures

In this section, the effect of 3D convolution on the performance of action recognition is analyzed. The length of the clips, the input modalities, distillation are some of the topics covered in this section. Additionally, popular 3D CNN architectures from the literature are analyzed not only in the perspective of performance but also the time complexity and memory utilization. All possible test types of 3D CNN architecture and some other details are given in Appendix C.2.

3.4.1 Effects of Clip Length and Input Resolution on the performance of 3D CNN Architectures

In this part, the main focus is on the effect of clip length. However, this section includes a brief study of input resolution as well. For this analysis, it is considered that 3D CNN versions of ResNet101 and ResNeXt101 architectures [24] are appropriate for studying the effect of clip length and input resolution. Additionally, in this sec-

tion, the effect of group convolution is also analyzed, since ResNeXt101 is the same version of ResNet101 with group convolution. The result of these studies is shown in Table 3.8.

There are two types of 3D ResNet backbone in Table 3.8, which are ResNet101 and ResNeXt101. 16f and 64f are to denote that 16 framed and 64 framed clips are used in training the architecture, respectively. "16f pre-trained" specified in 64f architectures is to denote that 64 framed clips are used in the fine-tuning of the HMDB51 but the pre-trained weights are obtained by training the architecture on Kinetics with 16 framed clips. The fine-tuning of the architectures are implemented with the Kinetics pre-trained weights released by the authors of the paper [24], and the same input sizes (112x112) and the same mean and standard deviation of the authors are applied. The only exception is "ResNeXt101-64f-224 (64f-112 pre-trained)" where the input resolution is set to 224x224 during the fine-tuning. The test results are obtained with multiple crops and multiple clips settings (See Section C.2).

The architecture changes from ResNet101-16f to ResNeXt101-16f and ResNet101-64f (with 16f pre-trained) to ResNeXt101-64f (with 16f pre-trained) increase Top-1 performance with about 2%, which shows the effectiveness of group convolution. Increasing the clip length from 16 to 64 during fine-tuning even though pre-trained weights are still obtained with the training of 16-framed clips on the Kinetics dataset increases the performance with about 5.5% increase in both ResNet101 and ResNeXt101 architectures. If both of the pre-training and fine-tuning are implemented with 64 frames, the performance reaches up to 73.07%, which is about 4.7% more compared to ResNeXt101-64f (16f pre-trained) and about 10% more compared to ResNeXt101-16f training. Increasing the input resolution from 112 to 224 by fine-tuning increases the top1 performance by about 2% and it is expected that increasing the input resolution from 112 to 224 also in pre-training would increase the performance even more but with the cost of computational complexity.

3.4.2 Two-stream 3D Architectures

Two stream architectures consist of two replicas of the same architectures, where one stream is fed by RGB, and the other stream is fed by optical flow. The information

Table 3.9: Results of Two-stream 3D architectures on HMDB51 Split-1

Method	**Modality**	**Top-1**	**Top-3**
ResNeXt101	RGB	73.07	90.20
ResNeXt101	Flow	79.80	91.63
ResNeXt101	**Two-Stream**	**82.35**	**94.38**
I3D	RGB	75.42	91.57
I3D	Flow	77.97	92.22
I3D	**Two-Stream**	82.03	93.99

about input modalities can be reached in Section 3.2.6. For the test of two-stream architectures, two architectures are selected for comparison. These are ResNeXt101 architecture (see Section 2.2.3.2) and I3D architecture (see Section 2.2.3.1). The implementation details of the architecture i given in Section 3.2. The only exception is that the I3D architectures are trained by the learning rate 10^{-1} instead of 10^{-2} because of the difficulty in the training.

Top-1 and Top-3 results of RGB, flow and two-stream of the ResNeXt101 and I3D architectures are tabulated in Table 3.9. In this table, the results of Top-1 and Top-3 results are given. The detailed view of this table is presented in Table C.1 in Appendix C.

For the comparison of RGB stream, I3D seems to be better than ResNeXt101 architecture. However, for flow stream and two-stream, ResNeXt101 architecture is better than I3D architecture. As can be observed from Table 3.9, the optical flow field yields better results compared to the RGB streams of the architecture, such that about a 6.7% increase in ResNeXt101 architecture and 2.5% increase in I3D architecture. The fact that the optical flow stream is better than the RGB stream is also related to the characteristics of the HMDB51 dataset, where severe camera motions do not exist. Comparably, the camera motion is more severe in the Kinetics dataset, resulting in low performances in the optical flow stream compared to the RGB stream [5].

In order to emphasize the importance of the dataset in pre-training, there is a need to highlight the difference of the fine-tuning results of I3D on HMDB51 split1 between the pre-trained Kinetics and pre-trained ImageNet. In the I3D paper [5], I3D-RGB, I3D-Flow, and I3D-two-stream obtain 49.8, 61.9, and 66.4, respectively. However, according to the result of Table 3.9, I3D-RGB, I3D-Flow, and I3D-two-stream obtain

Table 3.10: Performance and Parameter Size Comparison for RGB input modalities on HMDB51 split-1

Method	Top-1	Top-3	# Parameters	# Operations
ResNeXt101 112x112 16f	63.33	82.61	47.63 M	**9.64** GMac
ResNeXt101 112x112 64f	73.07	90.20	47.63 M	38.56 GMac
MFNET 224x224 16f	70.20	87.58	**7.73** M	11.25 GMac
Rep-Flow-50 224x224 32f	72.42	89.54	29.09 M	44.59 GMac
TSM-50 224x224 8f	66.80	87.25	23.61 M	32.96 GMac
TSM-50 224x224 8x8f	72.03	89.54	23.61 M	32.96 GMac
TSM-50 [3] 224x224 8x8f	73.79	90.13	23.61 M	32.96 GMac
Modified ResNet50 224x224 32x2f	71.44	88.69	27.33 M	33.05 GMac
Modified ResNet50 Non-local 224x224 32x2f	72.88	91.44	34.69 M	38.22 GMac
Modified ResNet50 224x224 64f	73.79	91.70	27.33 M	66.10 GMac
Modified ResNet50 Non-local 224x224 64f	73.27	90.26	34.69 M	76.43 GMac
I3D 224x224 64f	74.90	91.63	12.34 M	111.33 GMac
SlowFast-50 (8x8) 224x224 64f	78.37	92.68	33.76 M	50.72 GMac
MARS ResNext101 112x112 64f	80.72	92.75	47.63 M	38.56 GMac
R(2+1)D ResNet34 [4] 112x112 32f	**81.76**	**93.86**	63.52 M	152.95 GMac

75.42, 77.97, and 82.03, respectively.

3.4.3 Comparison of 3D CNN Architectures

[3] The selection procedure of frames in pre-training is as in [70]. Three splits top1 average is denoted as 73.5

In this section, various CNN architectures with RGB input modalities are compared. These are ResNext architectures (Section 2.2.3.2), MFNET (Section 2.2.8), Rep-Flow architecture [50], TSM architectures (Section 2.2.9), Modified ResNet architecture (Section 2.2.5), I3D architecture (Section 2.2.3.1), SlowFast architectures (Section 2.2.6), MARS architectures (Section 2.2.7) and R(2+1)D architectures (Section 2.2.4). It should be denoted that TSM is not a 3D convolution architecture but there is a temporal information transfer with shifting operations (Section 2.2.9 for more details).

Top-1 and Top-3 results on HMDB-51 split with RGB input modality is presented in Table 3.10. The results in Table 3.10 are calculated by using non-overlapping multiple clips and 10 crop settings. The detailed view of the table with all four different test settings is presented in Appendix-C Table C.2. For all of the architectures, Kinetics pre-trained weights released by the related GitHub repositories are utilized, except the R(2+1)D in which the IG-65M dataset is utilized (Section 3.1.5). The normalization parameters and input sizes are determined suitably with the pre-training schemes of the architecture. The implementation details explained in Section 3.2 are followed. The only difference is that for the training of Rep-Flow, the effective batch size is set to 24, and the learning rate of the flow layer is set to 1/100 of the learning rate of the other layers of the architecture as suggested in [50].

As indicated in the footnote [3], one of TSM architecture is pre-trained with a segment-based sampling of the frames as implemented in [71], instead of dense sampling strategy followed in the pre-training of other architectures. Another point is related to the number of frames in the selected clips. 64f is to denote that the clips consist of 64 frames. Additionally, 32x2f is to denote that the clips consist of 32 frames but frame selection is implemented with a stride of two, resulting in a 64 frame length coverage with 32 frames. 8x8f similarly covers 64 frame length.

It should be highlighted that Table C.2 indicates that the utilization of ten crops beneficial for nearly half of the architecture from the top1 performance perspective. From

% in [41]. The frame selection in fine-tuning and test seems ambiguous for me.

[4] The pre-training is implemented with IG-65M, while the pre-training of other methods are implemented with Kinetics-400. Therefore, the dataset has also effect on obtaining the best performance in the table. Top1 result of R(2+1)D with Kinetics pre-training is denoted as 74.4 % in [49]. Therefore, about 7.7 % increase seems to be the result of the change in pre-training dataset.

the Top-3 performance perspective, it can be concluded that utilization of ten crops is mostly beneficial but with the expense of ten times computational complexity. The utilization of multiple clips is always beneficial since it includes different parts of the action.

The best result in Table 3.10 is obtained with R(2+1)D ResNet34 architecture. However, it should be denoted that this architecture is pre-trained with IG-65 while the others are pre-trained with Kinetics 400. It should not be ignored that the dataset has a significant effect on the increase in the performance of this architecture. As indicated in the footnote [4], the performance improvement brought by IG-65M is about 6-7%. In MARS architecture, the significant benefits of the distillation concept are observed. As it is explained in Section 2.2.7, the only difference of MARS ResNeXt from the traditional ResNeXt is the training procedure in which the features of RGB architecture is distilled with the features of optical flow architecture. Therefore, during the inference, with the same computational complexity and memory consumption, the Top-1 performance of the architecture has increased with 5%. If the effect of the IG-65M dataset and the distillation are ignored, it might be possible to conclude that the best 3D architecture is SlowFast.

Another point that needs attention is the utilization of non-local blocks. For the results of the modified ResNet50 architectures, non-local has improved the performance of 32x2f, but not the 64f preference. In [72], non-local blocks are shown to increase the performance of not only the activity recognition tasks but also object detection and segmentation tasks. One of the possible reasons for not observing the performance improvement in 64f settings might be that the pre-training weights are obtained with 32x2f settings. Another conclusion from the results of modified ResNet50 architectures is that 64f architectures result in better performances although 32x2f and 64f architectures cover the same duration. Therefore, it can be concluded that even the utilization of skipped frames might be important.

For TSM architectures, with the same budget of the same number of frames, the sparse selection of frames is shown to results in significant performance increases, up to about 5% increase in Top-1 and 2% increase in Top-3 results from 8f to 8x8f implementation. Additionally, segment-based selection [3] in pre-training results in better

performance in the performance of TSM architecture, as indicated in the GitHub page of the paper.

For SlowFast networks, the benefits of parallel fast architecture are demonstrated clearly in Table 3.10. Despite that slow path is the same as the traditional ResNet architecture, the addition of the fast path and the lateral connection from fast to slow path boosts the performance significantly. The (8x8) of SlowFast architecture corresponds to $\alpha = 8$ and $\beta = \frac{1}{4}$ in architecture settings (See Section 2.2.6).

As an additional note, the ensemble of the best three architectures results in 85.62% and 95.82% Top-1 and Top-3 results on Split-1 of the HMDB51 dataset with single clip single crop settings. The top-1 result of the ensemble of the architectures is about 4% more than the best architecture in Table 3.10. Therefore, it can be concluded that different architectures might learn complementary information to each other. The ensemble of the architectures can be perceived as a different utilization of two-stream architectures in a sense that it is a three-stream architecture where streams are not identical to each other and which are fed with only RGB input.

3.4.4 Computational Complexity and Memory Utilization Analysis of the Architectures:

Apart from the recognition performances, the number of parameters and the number of operations are two other important factors for the selection of the architectures. For many practical applications of these methods, there might be some memory limitations or time considerations. The results of these analyses are shown in Table 3.10. In order to calculate the number of parameters and the number of operations a GitHub repo[5] is utilized. Denoting the number of operations, Multiply - Accumulate Operation (Mac) is utilized as a unit, which assumes $a * x + b$ as one operation according to the repo. Comparing the Mac unit with the FLOP unit in the literature [6, 19], it can be concluded that one Mac is equal to one FLOP.

It should be noted that in multiple clips tests (instead of single clip tests), the concept of the number of clips appears. This concept is important for a fair comparison. In

[5] github.com/sovrasov/flops-counter.pytorch

order to make it more clear; for instance, the number of clips for 16f and 64f to process a video with 64 frames are four and one, respectively.

There are some possible choices of giving clips as inputs to architectures. One possible way is to give them at once by increasing the batch size, resulting in a proportional increase in memory usage with the number of clips. Another way is giving them sequentially, resulting in a proportional increase in time complexity with the number of clips. A compromise between memory utilization and time complexity is also possible. If the number of clips factor is ignored, the comparison between the architectures with different frames would be unfair.

For instance, consider MFNET 16f and I3D 64f architectures. A video with 64 frames is handled with only one clip with I3D but with four clips with MFNET. Therefore, the three possible ways of MFNET would be 7.73x4 M parameters with the time complexity of 11.25 GMac, or 7.73x2 M parameters with the time complexity of 11.25x2 GMac, or 7.73M parameters with the time complexity of 11.25x4 GMac. A more realistic comparison in this table should not also ignore this factor. To perceive it from another perspective, there is a need to highlight the difference between time complexity and computational complexity. Changing the batch size of the architecture does not change the total computational complexity of architecture for a given video. However, there can be a trade-off between memory utilization and time complexity by changing the batch size. For instance, MFNET has 11.25x4 GMac computational complexity to get the result of 64 framed video.

Based on the results of Table 3.10, the architecture with the least number of parameters is MFNET 16f (7.73M) and the architecture with the least number of operations is ResNeXt 16f (9.64 GMac). However, ResNeXt Top-1 performance is about 8% lower compared to the MFNET architecture; therefore, 1.64 GMac computational increase for 8% Top1 performance increase seems to be a fair deal. As a result, it might be concluded that MFNET is one of the best efficient architecture of this table. However, the clip length factor of MFNET should not be forgotten while comparing with the other architectures.

For the performance of TSM architectures, it seems that the utilization of 2D CNN architectures does not provide significant benefits in the reduction of parameter size

and complexity when compared with MFNET or modified ResNet50 32x2f. However, one of the advantages of TSM architectures is that the early decision can be made directly with the first frame of the architecture, which decreases the number of operations to 32.96/8 GMac. Therefore, TSM can be utilized in any application where the low latency is crucial. However, the performance of the early frames is not investigated in this thesis work.

Aside, I3D has a very efficient memory utilization but its computational complexity is significantly worse compared to the other architectures, except the R(2+1)D ResNet34 architecture. For the comparison between SlowFast-50 and MARS architectures, SlowFast is better from the memory utilization perspective but worse in the computational complexity perspective. However, it should be highlighted that the utilization of 112x112 input size results in better computational complexity. Therefore another reason for the less computational complexity of ResNeXt from Inception I3D is the input size, not purely the architecture itself.

CHAPTER 4

PROPOSED METHOD: BERT ON 3D CNN ARCHITECTURES

Action Recognition (AR) pertains to identifying the label of the action or the activity observed in a video clip. With cameras everywhere, AR has become essential in many domains, such as video retrieval, surveillance, human-computer interaction, and robotics.

A video clip contains two critical pieces of information for AR: Spatial and temporal information. Spatial information represents the static information in the scene, such as objects, context, entities, etc., which are visible in a single frame of the video, whereas temporal information, obtained by integrating the spatial information over frames, mostly captures the dynamic nature of the action.

In this work, the joint utilization of two temporal modeling concepts from the literature, which are 3D convolution and late temporal modeling, is proposed and analyzed. Briefly, 3D convolution is a way of generating a temporal relationship hierarchically from the beginning to the end of CNN architectures. On the other hand, late temporal modeling is typically utilized with 2D CNN architectures, where the features extracted by 2D CNN architectures from the selected frames are usually modeled with recurrent architectures, such as LSTM, Conv LSTM.

Despite its advantages, the temporal global average pooling (TGAP) layer which is used at the end of all 3D CNN architectures [5, 6, 16, 24, 49, 63, 64, 76] hinders the richness of final temporal information. The features before TGAP can be considered as features of different temporal regions of a clip or video. Although the receptive field might cover the whole clip, the effective receptive field has a Gaussian distribu-

tion [44], producing features focusing on different temporal regions of a clip. In order to discriminate actions, one part of the temporal feature might be more important than the others or the order of the temporal features might be more beneficial than simply averaging the temporal information. Therefore, TGAP ignores this ordering and fails to fully exploit the temporal information.

Therefore, we propose using the attention mechanism of BERT for better temporal modeling than TGAP. BERT determines which temporal features are more important with its multi-head attention mechanism.

To the best of our knowledge, our work is the first to propose replacing TGAP in 3D CNN architectures with late temporal modeling. We also consider that this study is the first to utilize BERT as a temporal pooling strategy in AR. We show that BERT performs better temporal pooling than average pooling, concatenation pooling, and standard LSTM. Moreover, we demonstrate that late temporal modeling with BERT improves the performances of various popular 3D CNN architectures for AR which are ResNeXt101, I3D, SlowFast, and R(2+1)D by using the split-1 of the HMDB51 dataset. Using BERT R(2+1)D architecture, we obtain the new state of the art results; 85.10% and 98.69% Top-1 performances in HMDB51 and UCF101 datasets, respectively.

4.1 Proposed Methods

In this part, the proposed methods of this study are introduced. Firstly, the main proposed method, namely BERT-based temporal modeling with 3D CNN for activity recognition, is presented in Section 4.1.1. Next, some novel feature reduction blocks are proposed in Section 4.1.2. These blocks are utilized to reduce the number of parameters of the proposed BERT-based temporal modeling. Thirdly, the proposed BERT-based temporal modeling implementations on SlowFast architecture are examined in Section 4.1.3. The reason to re-consider the BERT-based late temporal modeling on SlowFast architecture is due to its different two-stream structure from other 3D CNN architectures.

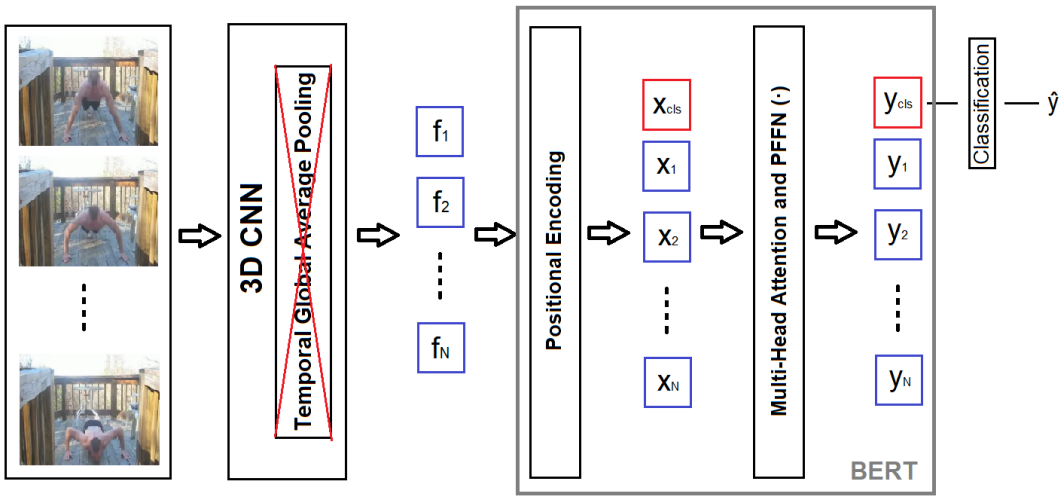


Figure 4.1: BERT-based late temporal modeling

4.1.1 BERT-based Temporal Modeling with 3D CNNs for Action Recognition

Bi-directional Encoder Representations from Transformers (BERT) [9] is a bidirectional self-attention method, which has provided unprecedented success in many downstream Natural Language Processing (NLP) tasks. The bidirectional property enables BERT to fuse the contextual information from both directions, instead of relying upon only a single direction, as in former recurrent neural networks or other self-attention methods, such as Transformer [65]. Moreover, BERT introduces challenging unsupervised pre-training tasks which leads to useful representations for many tasks.

Our architecture utilizes BERT-based temporal pooling as shown in Figure 4.1. In this architecture, the selected K frames from the input sequence are propagated through a 3D CNN architecture without applying temporal global average pooling at the end of the architecture. Then, in order to preserve the positional information, a learned positional encoding is added to the extracted features. In order to perform classification with BERT, additional classification embedding ($\mathbf{x_{cls}}$) is appended as in [9] (represented as red box in Figure 4.1). The classification of the architecture is implemented with the corresponding classification vector $\mathbf{y_{cls}}$ which is given to the fully connected layer, producing the predicted output label $\hat{y}$.

The general single head self-attention model of BERT as explained in Section 2.2.1

is formulated as:

$$\mathbf{y_i} = PFFN\left(\frac{1}{N(x)}\sum_{\forall j} g(\mathbf{x_j}) f(\mathbf{x_i}, \mathbf{x_j})\right), \tag{4.1}$$

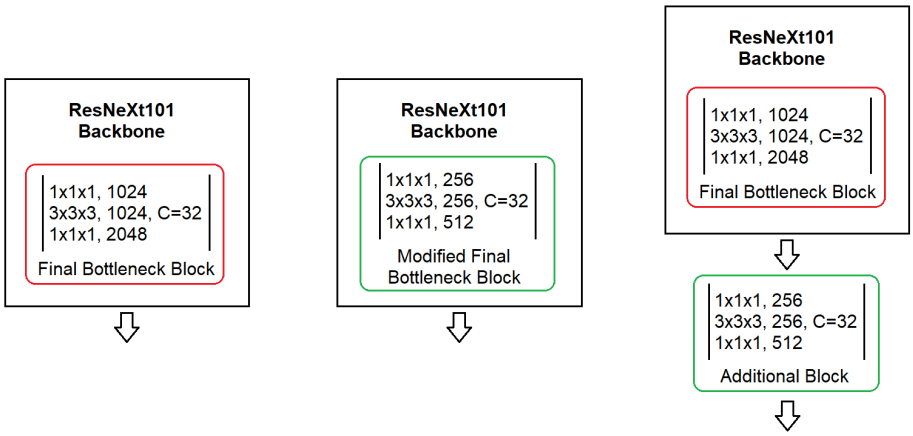
where $\mathbf{x_i}$ values are the embedding vectors that consists of extracted temporal visual information and its positional encoding; i indicates the index of the target output temporal position; j denotes all possible combinations; and $N(x)$ is the normalization term. Function $g(\cdot)$ is the linear projection inside the self-attention mechanism of BERT, whereas function $f(\cdot,\cdot)$ denotes the similarity between $\mathbf{x_i}$ and $\mathbf{x_j}$: $f(\mathbf{x_i}, \mathbf{x_j}) = \text{softmax}_j(\theta(\mathbf{x_i})^T \phi(\mathbf{x_j}))$, where the functions $\theta(\cdot)$ and $\phi(\cdot)$ are also linear projections. The learnable functions $g(\cdot)$, $\theta(\cdot)$ and $\phi(\cdot)$ try to project the feature embedding vectors to a better space where the attention mechanism works more efficiently. The outputs of $g(\cdot)$, $\theta(\cdot)$ and $\phi(\cdot)$ functions are also defined as *value*, *query* and *key*, respectively [65]. $PFFN(\cdot)$ is Position-wise Feed-forward Network applied to all positions separately and identically: $PFFN(x) = \mathbf{W_2} GELU(\mathbf{W_1 x} + \mathbf{b1}) + \mathbf{b2}$, where $GELU(\cdot)$ is the Gaussian Error Linear Unit (GELU) activation function [26].

The final decision of classification is performed with one more linear layer which takes $\mathbf{y_{cls}}$ as input. The explicit form of $\mathbf{y_{cls}}$ can be written as:

$$\mathbf{y_{cls}} = PFFN\left(\frac{1}{N(x)}\sum_{\forall j} g(\mathbf{x_j}) f(\mathbf{x_{cls}}, \mathbf{x_j})\right). \tag{4.2}$$

Therefore, our use of the temporal attention mechanism for BERT is not only to learn the convenient subspace where the attention mechanism works efficiently but also to learn the classification embedding which learns how to attend the temporal features of the 3D CNN architecture properly.

A similar work for action recognition is implemented with non-local neural networks (NN) [72]. The main aim of non-local block is to create global spatio-temporal relations, since convolution operation is limited to local regions. For this aim, non-local blocks use a similar attention concept by using 1x1x1 CNN filters, in order to realize $g(\cdot)$, $\theta(\cdot)$ and $\phi(\cdot)$ functions. The main difference between the non-local and the proposed BERT attention is that non-local concept [72] is preferred to be utilized not at the end of the architecture, but some preferred locations inside the architecture. However, our BERT-based temporal pooling is implemented on the extracted features of



(a) Original (b) FRMB (c) FRAB

Figure 4.2: The implementations of Feature Reduction with Modified Block (FRMB) and Feature Reduction with Additional Block (FRAB)

the 3D CNN architecture and utilizes multi-head attention concept to create multiple relations with self-attention mechanism. Moreover, it utilizes positional encoding in order to preserve the order information and utilizes learnable classification token.

Another similar study for action recognition is the video action transformer network [20] where the transformer is utilized in order to aggregate contextual information from other people and objects in the surrounding video. The video action transformer network deals with both action localization and action recognition; therefore, its problem formulation is different from ours and its attention mechanism needs to be reformulated for the late temporal modeling for action recognition. Differently from the video action transformer network, our proposed BERT-based late temporal modeling utilizes the learnable classification token, instead of using the pooled feature of the output of the backbone architecture.

4.1.2 Proposed Feature Reduction Blocks: FRAB & FRMB

The computational complexity of BERT has a quadratic increase with the dimension of the output feature of the CNN backbone. As a result, if the dimension of the output feature is not reduced for specific backbones, the BERT module might have more parameters than the backbone itself. For instance, if the dimension of the output

feature is 512, the single-layer BERT module has about 3 Million parameters, while the parameter size would be about 50 Million for the output feature dimension of 2048.

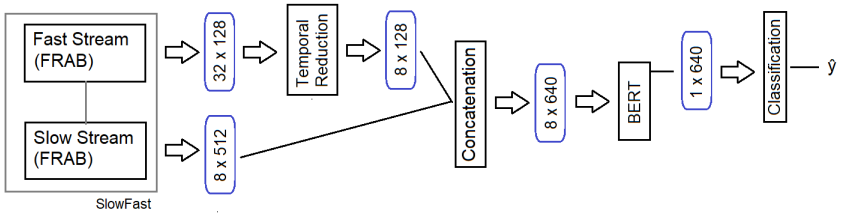
Therefore, in order to utilize BERT architecture in a more parameter efficient manner, two feature reduction blocks are proposed. These are Feature Reduction with Modified Block (FRMB) and Feature Reduction with Additional Block (FRAB). In FRMB, the final unit block of the CNN backbone is replaced with a novel unit block with the aim of feature dimension reduction. In FRAB, an additional unit block is appended to the backbone to reduce the dimension. An example implementation of FRMB and FRAB on ResNeXt101 backbone is presented in Figure 4.2.

The benefit of FRMB implementation is its better computational complexity and parameter efficiency over the FRAB implementation. Moreover, FRMB has even a better computational complexity and parameter efficiency than the original backbone. One possible downside of FRMB over FRAB is that the final block does not benefit from the pre-trained weights of the larger dataset if the feature reduction block is implemented only in the fine-tuning step but not in the pre-training.

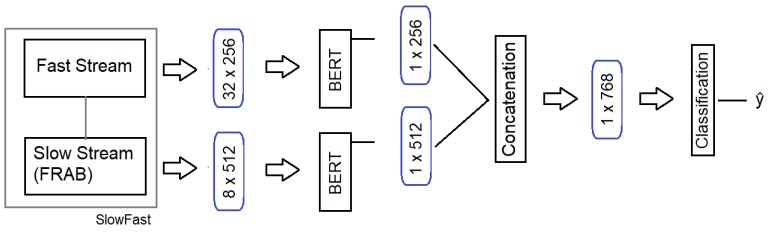
4.1.3 Proposed BERT Implementations on SlowFast Architecture

SlowFast architecture [16] introduces a different perspective for the two-stream architectures. Instead of utilizing two different modalities as two identical streams, the overall architecture includes two different streams (namely fast and slow streams or paths) with different capabilities for a single modality. In SlowFast architecture, the slow stream has a better spatial capability, while the fast stream has a better temporal capability. The fast stream has better temporal resolution and less channel capacity compared to the slow stream.

Due to its two-stream structure with different temporal resolutions, direct implementation of BERT-based late temporal modeling explained in Section 4.1.1 is not possible. Therefore, two alternative solutions are proposed in order to carry out BERT-based late temporal modeling on SlowFast architecture: Early-fusion BERT and late-fusion BERT. In early-fusion BERT, the temporal features are concatenated before



(a) Early-fusion



(b) Late-fusion

Figure 4.3: Early-fusion and late-fusion implementations of BERT on SlowFast architecture.

the BERT layer and only a single BERT module is utilized. To make the concatenation feasible, the temporal resolution of the fast stream is decreased to the temporal resolution of the slow stream. In late-fusion BERT, two different BERT modules are utilized, one for each stream and the outputs of two BERT modules from two streams are concatenated. The figure for early-fusion and late-fusion is shown in Figure 4.3.

4.2 Experimental Results

In this part, dataset, implementation details, ablation study, results on different architectures, and comparison with state-of-the-art sections are presented, respectively.

4.2.1 Dataset

For analyzing the improvements of BERT on individual architectures (Section 4.2.4), split 1 of the HMDB51 dataset is used, whereas the comparisons against the-state-of-the-art (See Section 4.2.5) are performed by using the three splits of the HMDB51 and UCF101 datasets. Additionally, the ablation study (See Section 4.2.3) is conducted

using the three splits of HMDB51. Moreover, Kinetics-400 and IG65M are used for pre-trained weights of the architectures before fine-tuning on HMDB51 and UCF101. The pre-trained weights are obtained from the authors of architectures, which are ResNeXt, I3D, Slowfast, and R(2+1)D. Among these architectures, R(2+1)D is pre-trained with IG65M but the rest of the architectures are pre-trained with Kinetics-400.

4.2.2 Implementation Details

For the standard architectures (with TGAP and without any modification to architectures), SGD with learning rate 10^{-2} is utilized, except I3D in which the learning rate is set to 10^{-1} empirically. For architectures with BERT, the ADAMW optimizer [43] with learning rate 10^{-5} is utilized except I3D for which the learning rate is set to 10^{-4} empirically. For all training runs, the “reducing learning rate on the plateau" schedule is followed. The data normalization schemes are selected conforming with the data normalization schemes of the pre-training of the architectures in order to benefit fully from pre-training weights. A multi-scale cropping scheme is applied for fine-tuning and testing of all architectures [69]. In the test time, the scores of non-overlapping clips are averaged. The optical flow of the frames is extracted with the TV-L1 algorithm (Appendix A).

In the BERT architecture, there are eight attention heads and one transformer block. The dropout ratio in $PFFN(\cdot)$ is set to 0.9. Mask operation is applied with 0.2 probability. Instead of using a mask token, the attention weight of the masked feature is set to zero. The classification token ($\mathbf{x_{cls}}$) and the learned positional embeddings are initialized as the zero-mean normal weight with 0.02 standard deviation. Default Torch linear layer initialization is used. Different from the I3D-BERT architecture, the linear layers of BERT are also initialized as the zero-mean normal weight with 0.02 standard deviation since it yields better results for I3D-BERT.

4.2.3 Ablation Study

In this section, we will analyze each step of our proposals and examine how our method compares with alternative pooling strategies (see Table 4.1). In this analysis,

Table 4.1: Ablation Study of RGB ResNeXt101 architecture for temporal pooling analysis on HMDB51. FRMB: Feature Reduction with Modified Block.

Type of Temporal Pooling	**FRMB**	**Optimizer**	**Top1 (%)**	**# of Params**	**# of Operations**
Average Pooling (Baseline)		SGD	74.46	47.63 M	38.56 GFlops
Average Pooling		ADAMW	75.99	47.63 M	38.56 GFlops
Average Pooling	✓	ADAMW	74.97	44.22 M	38.36 GFlops
Concatenation	✓	ADAMW	76.49	44.30 M	38.36 GFlops
LSTM	✓	ADAMW	74.18	47.58 M	38.37 GFlops
Concatenation + Fully Connected Layer	✓	ADAMW	76.84	47.45 M	38.36 GFlops
Non-local + Concatenation + Fully Connected Layer	✓	ADAMW	76.36	47.35 M	38.43 GFlops
BERT pooling (Ours)	✓	ADAMW	**77.49**	47.38 M	38.37 GFlops

the ResNeXt101 backbone is utilized with the RGB modality, with a 112x112 input image size, and with 64-frame clips. In Table 4.1, temporal pool types, the existence of Feature Reduction with Modified Block (FRMB), the type of the optimizer, top1 performances, the number of parameters, and the number of operations are presented as the columns of the analysis.

One important issue is the optimizer. For training BERT architectures in NLP tasks, the ADAM optimizer is usually selected [9]. However, SGD is preferred for 3D CNN architectures [24, 5, 16, 64, 8]. Therefore, for training BERT, we select ADAMW (i.e. not ADAM), since ADAMW improves the generalization capability of ADAM [43]. In this ablation study, ResNeXt101 architecture (with Average Pooling in Table 4.1) is also trained with ADAMW in Table 4.1 which shows 1.5% increase in performance compared to SGD.

In this ablation study, FRMB implementation is selected for two reasons over FRAB. Firstly, FRMB yields about 0.5% better top1 performance than FRAB. Secondly, FRMB has better computational complexity and parameter efficiency than FRAB. From the experiments of the ablation study, it is observed that FRMB has lower computational complexity and better parameter efficiency at the cost of $\sim$1% decrease in Top-1 performance compared to the standard backbone (Table 4.1). The impact

of FRMB on 2D CNN architectures with BERT-based late temporal modeling is explained in Appendix C.4.

For a fair comparison, we set the hyper-parameters of the other pooling strategies (LSTM, concatenation + fully connected layer, and non-local + concatenation + fully connected layer) such that the number of parameters and the number of operations of these temporal pooling strategies is almost the same compared to the proposed BERT pooling. LSTM is implemented in two stacks and with a hidden-layer size 450. The dimension of the inter-channel of a non-local attention block (the dimension size of the attention mechanism) is set equal to the input size to the non-local block which is 512. The number of nodes of a fully connected layer is determined according to the need for equal parameter size with the proposed BERT temporal pooling for a fair comparison.

When Table 4.1 is analyzed, one can observe that among the six different alternatives (with FRMB), BERT has the best temporal pooling strategy. Additionally, the proposed FRMB-ResNeXt101-BERT provides 3% better Top-1 accuracy than the ResNeXt101 average pooling (Baseline) despite the fact that FRMB-ResNeXt101-BERT has better computational complexity and parameter efficiency compared to ResNeXt101 average pooling (Baseline) (see Table 4.1). The BERT layer itself has about 3M parameters and negligible computational complexity with respect to the ResNeXt101 backbone. For the other temporal pooling strategies, LSTM worsens the performance with respect to the temporal average pooling. Concatenation and concatenation + fully connected layer are also other successful strategies in order to utilize the temporal features better than the average pooling. The addition of a non-local attention block before the concatenation + fully connected layer also decreases the performance compared to only concatenation + fully connected layer pooling implementation. It should be highlighted that the original implementation of the non-local study [72] also prefers not to utilize the non-local block at the end of the final three bottleneck blocks, which is a consistent fact with the experimental result of this study related with non-local implementation.

In addition, the ablation study of BERT late temporal modeling is performed and presented in Table 4.2. These results examine the effects of the number of layers, the

Table 4.2: Ablation Study of BERT late temporal Modeling on HMDB51.

Number of BERT Layers	Number of Attention Heads	Learnable Classification Token against Pooled Features	Top1 (%)
1	8		76.07
1	1	✓	76.97
1	8	✓	**77.49**
2	8	✓	77.24

number of heads, and utilization of learnable classification token instead of the average pooled feature. Initially, the experiment of replacing the average of extracted temporal features with learnable classification token results in a 1.42% Top-1 accuracy boost. Next, utilization of multi-head attention with eight attention heads improves the Top-1 performance of single-head attention with 0.52%. Thirdly, increasing the number of layers from one to two worsens the top1 performance with 0.25%. Moreover, the memory trade-off of every layer of BERT is about 3M. The reason behind the deterioration might be the fact that late temporal modeling is not as much complex as capturing rich linguistic information and a single layer might be enough to capture the temporal relationship between the output features of 3D CNN architectures.

4.2.4 Results on Different 3D CNN Architectures

In this section, the improvements obtained by replacing TGAP with BERT pooling on popular 3D convolution architectures for action recognition is presented, including ResNeXt101 [24], I3D [5], SlowFast[16] and R(2+1)D [64].

4.2.4.1 ResNeXt Architecture

ResNeXt architecture is essentially ResNet with group convolutions [24]. For testing this architecture, the input size is selected as 112x112 as in the study of [24, 8] and 64 frame length is utilized.

The results of the ResNeXt101 architecture is presented in Table 4.3. The performance of the architectures is compared over RGB modality, (optical) flow modality, and both (two-stream) in which both RGB and flow-streams are utilized, and the

Table 4.3: Analysis of ResNeXt101 architecture with and without BERT for RGB, Flow, and two-stream modalities on HMDB51 split-1

BERT	Modality	Top-1	# Parameters	# Operations
	RGB	74.38	47.63 M	38.56 GFlops
✓	RGB	**77.25**	47.38 M	38.37 GFlops
	Flow	79.48	47.60 M	34.16 GFlops
✓	Flow	**82.03**	47.36 M	33.97 GFlops
	Both	82.09	95.23 M	72.72 GFlops
✓	Both	**83.99**	94.74 M	72.34 GFlops

scores are summed from each stream. In this table, the number of parameters and operations of the architectures are also presented. For feature reduction, FRMB is chosen instead of FRAB and its reasoning is explained in Section 4.2.3. (see Section 4.1.2 for more details about FRAB and FRMB). Based on the results in Table 4.3, the most important observation is the improvement of the performance by using BERT over the standard architectures (without BERT) in *all* modalities.

4.2.4.2 I3D Architecture

I3D architecture is an Inception-type architecture. During I3D experiments, the input size is selected as 224x224 and 64 frame length is used conforming with the I3D study [5]. The result of BERT experiments on I3D architecture is given in Table 4.4. In this table, there are two BERT implementations that are with and without FRAB. For I3D-BERT architectures with FRAB, the final feature dimension of the I3D backbone is reduced from 1024 to 512 in order to utilize BERT in a more parameter efficient manner. However, contrary to the ResNeXt101-BERT architecture, FRAB is selected instead of FRMB, because FRAB obtains about 3.6% better Top-1 result for RGB-I3D-BERT architecture on split1 of HMDB51.

The experimental results in Table 4.4 indicate that BERT increases the performance of I3D architectures in all modalities. However, the increase in RGB modality is quite limited. For the Flow modality, although there is a performance improvement for BERT without FRAB, the implementation of BERT with FRAB performs worse than the standard I3D architecture, implying that preserving the feature size is more important for flow modality compared to RGB modality in I3D architecture. For

Table 4.4: The performance analysis of I3D architecture with and without BERT for RGB, Flow, and two-stream modalities on HMDB51 split-1

BERT	**Modality**	**Feature Reduction**	Top-1	**# Parameters**	**# Operations**
	RGB	X	75.42	12.34 M	111.33 GFlops
✓	RGB	FRAB	**75.75**	16.40 M	111.72 GFlops
✓	RGB	X	75.69	24.95 M	111.44 GFlops
	Flow	X	77.97	12.32 M	102.52 GFlops
✓	Flow	FRAB	77.25	16.37 M	102.91 GFlops
✓	Flow	X	**78.37**	24.92 M	102.63 GFlops
	Both	X	82.03	24.66 M	213.85 GFlops
✓	Both	FRAB	**82.68**	32.77 M	214.63 GFlops
✓	Both	X	**82.68**	49.87 M	214.07 GFlops

the two-stream setting, both of the proposed BERT architectures perform equally with each other and perform better than standard I3D with 0.65% Top-1 performance increase. Comparing with the ResNeXt101 architecture, the performance improvements brought by BERT temporal modeling is lower in I3D architecture.

4.2.4.3 SlowFast Architecture

The SlowFast architecture in these experiments is derived from a ResNet-50 architecture. The channel capacity of the fast streams is one-eighth of the channel capacity of the slow stream. The temporal resolution of the fast stream is four times the temporal resolution for the slow stream. The input size is selected as 224x224 and 64-frame length is utilized with the SlowFast architecture conforming with the SlowFast study [16]. Although it might be possible to utilize SlowFast architecture with also optical flow modality, the authors of SlowFast did not consider this strategy in their study. Therefore, in this effort, the analysis of BERT is also implemented by only considering the RGB modality.

In order to utilize BERT architecture with fewer parameters, the final feature dimension of SlowFast backbone is reduced similar to the ResNeXt101-BERT and I3D-BERT architectures. Similar to the I3D-BERT architecture, FRAB is chosen instead of FRMB since FRAB obtains about 1.5% better Top-1 result for SlowFast-BERT architecture on the split1 of HMDB51 (see Section 4.1.2 for more details about FRAB

Table 4.5: The performance analysis of SlowFast architecture with and without BERT for RGB modality on HMDB51 split-1

BERT	Top-1	# Parameters	# Operations
	79.41	33.76 M	50.72 GFlops
✓(*early-fusion*)	79.54	43.17 M	52.39 GFlops
✓(*late-fusion*)	**80.78**	42.04 M	52.14 GFlops

and FRMB). For early-fusion BERT, the feature dimension of the slow stream is reduced from 2048 to 512 and the feature dimension of the fast stream is reduced from 256 to 128. For late-fusion BERT, only the feature dimension of the slow stream is reduced from 2048 to 512. The details about the size of the dimensions are presented in Figure 4.3. The proposed implementation of BERT-based late temporal modeling on SlowFast architecture is presented in Section 4.1.3.

The results for BERT on SlowFast architecture are given in Table 4.5. First of all, both BERT solutions perform better than the standard SlowFast architecture but the improvement of early-fusion method is quite limited. Late-fusion BERT improves the top1 performance of standard SlowFast architecture with about 1.3 %. From the number of parameters perspective, the implementation of BERT on SlowFast architecture is not as much as efficient in comparison to ResNeXt101 architecture because of the FRAB implementation instead of FRMB as in the case of I3D-BERT. Moreover, the parameter increase of RGB-SlowFast-BERT is even higher than RGB-I3D-BERT because of the two-stream implementation of SlowFast network for RGB input modality. The increase in the number of operations is also higher in the implementation of SlowFast-BERT than the I3D-BERT and ResNeXt101-BERT because of the higher temporal resolution in SlowFast architecture and two-stream implementation for RGB modality.

For the two alternatives proposed BERT solution in Table 4.5, late-fusion yields better performance with better computational complexity in contrast with early-fusion BERT. Although the attention mechanism is implemented jointly on the concatenated features, the destruction of the temporal richness of fast stream to some degree might be the reason for the inferior performance of the early-fusion BERT.

Table 4.6: The performance analysis of R(2+1)D architecture with and without BERT for RGB modality on HMDB51 split-1

BERT	Top-1	# Parameters	# Operations
	82.81	63.67 M	152.95 GFlops
✓	**84.77**	66.67 M	152.97 GFlops

4.2.4.4 R(2+1)D Architecture

R(2+1)D [64] architecture is a ResNet-type architecture consisting of separable 3D convolutions in which temporal and spatial convolutions are implemented separately. For this architecture, 112x112 input dimensions are applied following the paper, and 32-frame length is applied instead of 64-frame because of the huge memory demand of this architecture and to be consistent with the paper [64]. The selected R(2+1)D architecture has 34 layers and implemented with basic block type instead of bottleneck block type (for further details about block types, see [24]). The most important difference of R(2+1)D experiments from the previous architectures is the utilization of the IG65M pre-trained weights, instead of Kinetics pre-trained weights (see Section 4.2.1 for details). Therefore, this information should always be considered while comparing this architecture with the aforementioned ones. The analysis of R(2+1)D BERT architecture is limited to RGB modality, since the study [19] of the IG65M dataset where R(2+1)D architecture is preferred is limited to RGB modality.

The experiments for BERT on R(2+1)D architecture are presented in Table 4.6. The feature dimension of R(2+1)D architecture is already 512 which is the same with the reduced feature dimension of ResNeXt101 and I3D backbones for BERT implementations. Therefore, we do not use FRMB or FRAB for R(2+1)D. There is an increase of about 3M parameters and the increase in the number of operations is still negligible. The performance increase of BERT on R(2+1)D architecture is about 2% which is a significant increase for RGB modality as in the case of ResNeXt101-BERT architecture.

4.2.5 Comparison with State-of-the-Art

In this section, the results of the best BERT architectures from the previous section are compared against the state-of-the-art methods. For this aim, two leading BERT architectures are selected among all the test methods: Two-Stream BERT ResNeXt101 and RGB BERT R(2+1)D (see Section 4.2.4). Note that these two architectures use different pre-training datasets, namely IG65 and Kinetics-400 for ResNext101 and R(2+1)D, respectively.

The results of the architectures on HMDB51 and UCF101 datasets are presented in Table 4.7. The table indicates if an architecture employs explicit optical flow. Moreover, the table lists the pre-training dataset used by the methods.

As shown in Table 4.7, BERT increases the Top-1 performance of the two-stream ResNeXt101 with 1.77% and 0.41% in HMDB51 and UCF101, respectively. Additionally, BERT improves the Top-1 performance of RGB R(2+1)D (32f) with 3.5 % and 0.48% in HMDB51 and UCF101, respectively, where 32f corresponds to 32-frame length. The results obtained by the R(2+1)D BERT (64f) architecture pre-trained with the IG65M dataset is the current state-of-the-art result in AR for HMDB51 and UCF101, to the best of our knowledge.

Among the architectures pre-trained in Kinetics-400, the two-stream ResNeXt101 BERT is again the best in HMDB51 but the second-best in the UCF101 dataset. This might be owing to the fact that HMDB51 involves some actions that can be resolved only using temporal reasoning and therefore benefits from BERT's capacity.

An important point to note from the table is the effect of pre-training with the IG65M dataset. RGB R(2+1)D (32f) (without Flow) pre-trained with IG65M obtains about 6% and 1.4% better Top-1 performance in HMDB51 and UCF101, respectively than the one pre-trained with Kinetics-400, indicating the importance of the number of samples in the pre-training dataset even if the samples are collected in a weakly-supervised manner.

Table 4.7: Comparison with the state-of-the-art.

Model	Uses Flow?	Extra Training Data	HMDB51	UCF101
IDT [67]	✓		61.70	-
Two-Stream [57]	✓	ImageNet	59.40	88.00
Two-stream Fusion + IDT [17]	✓	ImageNet	69.20	93.50
ActionVlad + IDT [22]	✓	ImageNet	69.80	93.60
TSN [71]	✓	ImageNet	71.00	94.90
RSTAN + IDT [13]	✓	ImageNet	79.90	95.10
TSM [41]		Kinetics-400	73.50	95.90
R(2+1)D [64]		Kinetics-400	74.50	96.80
R(2+1)D [64]	✓	Kinetics-400	78.70	97.30
I3D [5]	✓	Kinetics-400	80.90	97.80
MARS + RGB + Flow [8]	✓	Kinetics-400	80.90	**98.10**
FcF [50]		Kinetics-400	81.10	-
ResNeXt101	✓	Kinetics-400	81.78	97.46
EvaNet [49]	✓	Kinetics-400	82.3	-
HAF+BoW/FV halluc [68]		Kinetics-400	82.48	-
ResNeXt101 BERT **(Ours)**	✓	Kinetics-400	**83.55**	97.87
R(2+1)D (32f)		IG65M	80.54	98.17
R(2+1)D BERT (32f) **(Ours)**		IG65M	83.99	98.65
R(2+1)D BERT (64f) **(Ours)**		IG65M	**85.10**	**98.69**

4.3 Discussion

This study combines the two major components from AR literature, namely late temporal modeling and 3D convolution. Although there are many pooling, fusion, and recurrent modeling strategies that are applied to the features from 2D CNN architectures, we firmly believe that this manuscript is the first study that removes temporal global average pooling (TGAP) and better employs temporal information at the output of 3D CNN architectures. To utilize these temporal features, an attention-based mechanism namely BERT is selected. The effectiveness of this idea is proven on most of the popular 3D CNN architectures which are ResNeXt, I3D, SlowFast, and R(2+1)D. In addition, significant improvements over the-state-of-the-art techniques are obtained in HMDB51 and UCF101 datasets.

CHAPTER 5

PROPOSED METHOD : BERT DISTILLATION

The superiority of BERT in Natural Language Processing (NLP) tasks is substantially related to its unsupervised pre-training procedure. During this pre-training, BERT architecture is guided to predict some of the masked words which increase its learning capacity for creating a relationship between the words with its attention mechanism. For this aim, we aim to discover the potential of unsupervised training of BERT architecture on the action recognition task by using the distillation concept. In this chapter, the joint utilization of distillation and unsupervised training concepts are shown to be beneficial even with the small action recognition dataset HMDB51.

5.1 Methodology

In NLP tasks, training of BERT architectures is implemented in two steps, which are pre-training and fine-tuning. In the pre-training part, the aim is to train the architecture to predict the words which are masked. Then, these pre-trained weights of BERT are utilized in other NLP tasks in a supervised manner, such as sentiment analysis or question-answer problems.

In this study, unsupervised training of BERT architecture is utilized in a slightly different manner and combined with the distillation concept. The distillation concept is initially utilized in MARS [8] architecture in order to transfer knowledge of flow architecture to RGB architecture in the action recognition task. This approach is shown to be quite useful. Similar to this approach, instead of implementing the similarity loss and classification loss separately as two different phases of training procedure of

BERT, they are combined into a single loss function in BERT distillation (5.1)

$$\text{Loss}(s, \hat{y}, \mathbf{y}, \mathbf{z}) = \text{CrossEntropy}(s, \hat{y}) + \lambda \sum_{i=1}^{N} ||\mathbf{y_i} - \mathbf{z_i}||^2, \quad ||\mathbf{y_i}|| = ||\mathbf{z_i}|| = 1 \quad (5.1)$$

where s is the ground truth classification output, y_is are the output of the BERT architecture and i denotes the temporal index as seen in Figure 5.1, and $\hat{y}$ is the predicted classification label obtained from y_{cls} feature which can be seen in Figure 5.1 . From the distillation perspective, BERT architecture is student architecture. z_is are the outputs of the teacher 3D CNN architecture without temporal global average pooling and characteristically similar to the outputs f_is. This teacher architecture can be any architecture that is typically considered to be beneficial to improve the performance of student BERT architecture, such as flow version of the student architecture or RGB architecture with better capacity than student BERT architecture. It should be emphasized that the teacher architecture is pre-trained before the implementation of BERT distillation and weight update is not applied to teacher architecture during the distillation training of BERT architecture.

5.2 Experimental Results

In this section, the experiments of the concept of distillation with unsupervised training of BERT architecture is presented. In the first step, there is a necessity of determining the value of λ in (5.1). For this purpose, the student architecture is selected as RGB-ResNeXt-101-BERT with FRMB, whereas the teacher architecture is selected as Flow-ResNeXt-101-BERT with FRMB. The result of this analysis is presented in Table 5.1. For the selection of the best result, we pick the best value of λ according to Top-1 + Top-3 score. The reason to follow such an approach is to reduce the noisy results of the specified λ. The better analysis would be the three-split-average results but we do not want to follow it due to the computational complexity perspective.

The results of the distillation tests are presented in Table 5.2. The upper part of the table belongs to result of our proposed BERT distillation and the below part of the table belongs to MARS distillations.

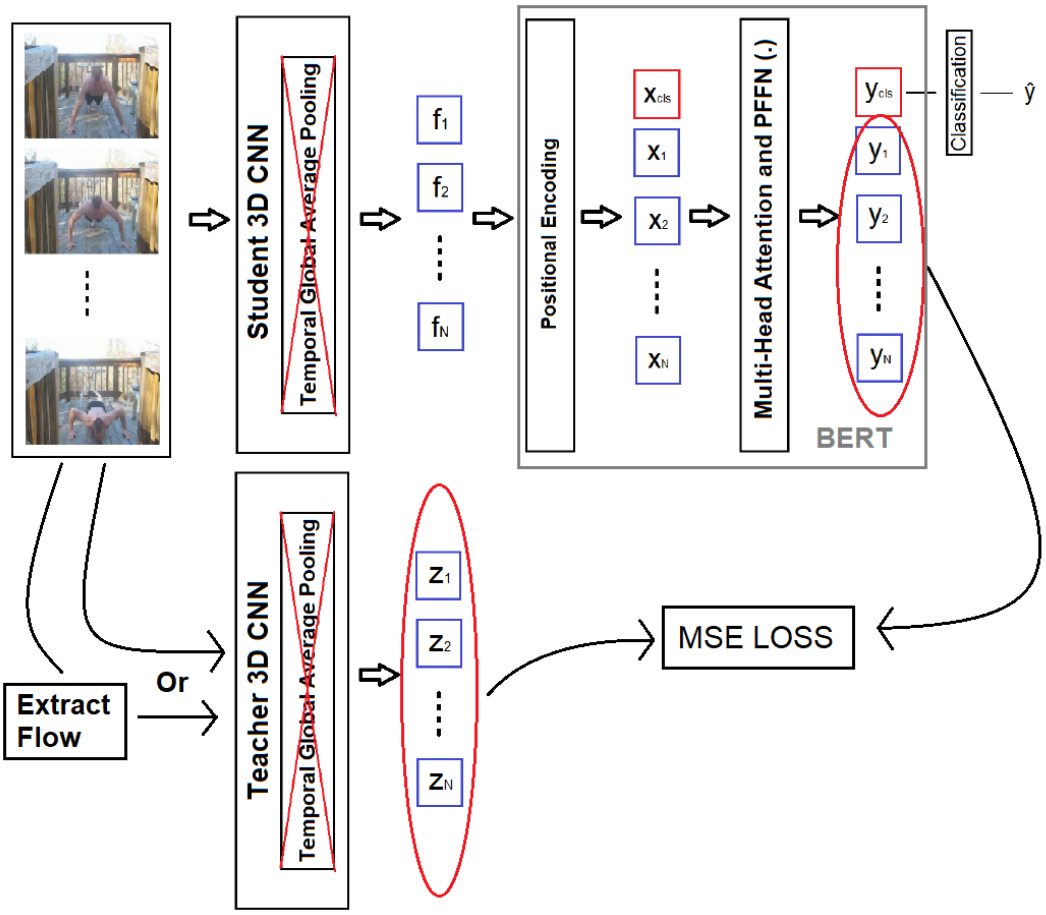


Figure 5.1: BERT-based Distillation

In the BERT distillation part, two architectures are selected as RGB-ResNeXt101-BERT FRMB and RGB-ResNeXt101-BERT FRAB. For the teacher architectures, Flow-ResNeXt101-BERT FRMB and RGB-R(2+1)D-BERT-(32x2f) are chosen and their Top-1 performances are 81.29% and 83.03%, respectively. Firstly, it can be observed that distillation from Flow-ResNeXt101-BERT-FRMB improves the performance of both RGB-ResNeXt101-BERT FRMB and RGB-ResNeXt101-BERT FRAB with about 0.8% and 1.6%, respectively. In addition, RGB-ResNeXt101-BERT-FRAB is distilled as a student architecture with Flow-ResNeXt101-BERT-FRMB and RGB-R(2+1)D-BERT-(32x2f) and Top-1 performances are improved with 1.6% and 1.0%, respectively. Although Top-1 performance of RGB-R(2+1)D-BERT is higher than Flow-ResNeXt101-BERT-FRMB, the latter is better, when it is used as a teacher architecture. This might be resulted due to the fact that different modality might contain more complementary information.

Table 5.1: Lambda parameter selection for distillation with unsupervised training of BERT architecture on split-1 of HMDB51

lambda (λ)	**Top-1**	**Top-3**	**Top-1 + Top-3**
25k	77.45	91.50	168.95
50k	78.69	91.11	169.80
100k	78.17	92.22	**170.39**
250k	78.37	91.44	169.81
500k	78.63	91.18	169.81

A comparison between MARS and BERT distillations can be stated based on the results of two (i.e. above and below) parts of Table 5.2. The most important difference between them is as follows: MARS distillation affects the features of classification directly, while BERT distillation affects indirectly because the distilled parts of BERT is not utilized directly in the classification layer. When MARS distillation is implemented in HMDB51 directly, it decreases the performance due to overfitting. However, such an overfitting problem is not observed during BERT distillation. This result might be observed because of the difference in the direct-indirect impact of BERT and MARS distillations. However, it is also shown that when MARS distillation is implemented by using the Kinetics dataset, the over-fitting problem disappears due to a large number of samples in the dataset and Top-1 accuracy improves significantly.

Table 5.2: Distillation with unsupervised training of BERT architectures and MARS distillations on HMDB51

Student Architecture	**Teacher Architecture**	**Top-1**	**Distillation Dataset**
RGB-ResNeXt101-BERT-FRMB	(No Distillation)	77.49	
RGB-ResNeXt101-BERT-FRAB	(No Distillation)	77.10	
RGB-ResNeXt101-BERT-FRMB	Flow-ResNeXt101-BERT-FRMB	78.30	HMDB51
RGB-ResNeXt101-BERT-FRAB	Flow-ResNeXt101-BERT-FRMB	**78.72**	HMDB51
RGB-ResNeXt101-BERT-FRAB	RGB-R(2+1)D-BERT-(32x2f)	78.15	HMDB51
RGB-ResNeXt101	(No Distillation)	74.46	
RGB-ResNeXt101 MARS	Flow-ResNeXt101	71.15	HMDB51
RGB-ResNeXt101 MARS	Flow-ResNeXt101	**81.22**	Kinetics

CHAPTER 6

SUMMARY & CONCLUSION

6.1 Summary

This thesis proposes novel neural network architectures for AR problem. Two recent techniques, namely BERT and distillation, are applied to conventional 3D CNNs in order to achieve late temporal modeling for the recognition task.

2D CNN architectures are presented in Section 3.3, starting with late temporal pooling strategies being examined on ResNet18 architecture with RGB modality in Section 3.3.1. Such late temporal modeling techniques cover the following: No-pooling, average pooling, non-local block + average pooling, concatenation, non-local block + concatenation, LSTM, Convolutional GRU, and BERT structures. Then, the impact of the additional fusion of intermediate features of 2D CNN architectures to final feature representation is analyzed in Section 3.3.2. Additionally, the impact of the depth of the CNN architecture is investigated for RGB modality in Section 3.3.3. Continuing this section, the impact of other input modalities, such as Flow and Pose, and joint utilization of all two modalities (RGB + Flow) and three modalities (RGB + Flow + Pose) are examined. For flow and pose modality, the impact of late temporal and fusion is also investigated in less detail compared to the analysis on RGB modality. It should be emphasized that the utilization of pose information is different from the literature; i,e. pose information is converted into RGB image which is explained in detail in Section 3.2.6).

About 3D CNN architectures (Section 3.4), some of the basic notions, such as the effect of clip length and input image resolution are initially investigated. The (pos-

itive) impact of group convolution is analyzed by comparing ResNet and ResNeXt. Additionally, the differences between applying these notions to both pre-training and fine-tuning, and only fine-tuning is compared. Next, two-stream architectures (RGB + Flow) is investigated by using ResNeXt and I3D architectures. Finally, the performance, memory utilization and computational complexity analysis of 3D CNN architectures are examined in Section 3.4.3 and Section 3.4.4. This analysis covers the architectures of ResNext, MFNET, Rep-Flow, TSM (Not 3DCNN but novel 2D CNN idea), Modified ResNet, I3D, SlowFast, MARS, and R(2+1)D architectures.

In this thesis, two novel solutions are proposed. In the first proposed method, which is presented in Chapter 4, a combination of late temporal modeling is approached from the perspective of 3D CNN architectures. However, the focus of this section is on using BERT as a late temporal pooling strategy. In this part of the thesis, BERT is compared against some of the late temporal pooling strategies, which are average pooling, LSTM, concatenation, concatenation + fully connected layer, non-local + concatenation + fully connected layer. In this chapter, in order to utilize BERT architecture in a more parameter efficient way, feature reduction methods (FRMB and FRAB) are proposed. The positive impact of BERT as a late temporal pooling strategy is shown on the architectures of ResNeXt, I3D, SlowFast, and R(2+1)D architectures. Moreover, for SlowFast architecture, two possible implementations are proposed for the implementation of BERT.

In the second proposed method, BERT is approached from the distillation concept perspective and the information of one architecture is transferred to another architecture by using the unsupervised training concept of BERT.

6.2 Conclusion

To begin with, one of the most important contributions of this thesis is the utilization of late temporal modeling for AR. In this thesis, late temporal modeling is covered for both 2D and 3D CNN architectures. Firstly, the success of the late temporal modeling strategy is dependent on both architecture and the modality. For instance, BERT is a better strategy than concatenation pooling for RGB 3D-ResNeXt101 but

vice versa for RGB 2D-ResNet18. On the other hand, BERT is a better strategy than concatenation pooling for flow and pose on 2D-ResNet18, but vice versa for RGB 2D-ResNet18. Additionally, one can argue that BERT is a better strategy than standard LSTM, average pooling, convolutional GRU. Moreover, fusing the final extracted features of 2D CNN architectures with its intermediate features demonstrates positive performance for both pose and RGB modality.

Another important conclusion is based on the comparison of 3D CNN architectures. The architectures are compared from the perspective of Top-1 accuracy, number of parameters, and number of operations. It can be concluded that from Top-1 accuracy perspective, SlowFast-50 is the best architecture among ResNeXt101, MFNET, Rep-Flow-50, TSM-50 (Not 3D CNN), Modified ResNet-50 [1], I3D, and R(2+1)D [2]. Besides, MFNET is a parameter efficient architecture, and the implementations of ResNeXt (64f), TSM (8x8f), and Modified ResNet-50 (32x2f) are good architectures from a computational complexity perspective.

On the other hand, some of the factors, which increase the performance of 3D CNN architectures, are determined by using split-1 of HMDB51 with RGB modality. Increasing the clip length from 16 to 64 increases the Top-1 accuracy of ResNeXt about 10%, whereas increasing the input resolution from 112 to 224 only in fine-tuning increases Top-1 accuracy of ResNeXt by about 2%. The implementation of group convolution (change from ResNet to ResNeXt) increases Top-1 accuracy by around 2%. The utilization of the MARS distillation concept on ResNeXt101 architecture increases Top-1 accuracy with about 7%. Pre-training of R(2+1)D architecture with IG65M dataset increases the performance of the one with Kinetics-400 increases the performance with about 6%, indicating that the scale of the pre-training dataset is also important. The concept of BERT distillation is proposed and it improves Top-1 performance of RGB-ResNeXt-BERT by 1.6%. The improvements of distillation and pre-training dataset are important from the perspective that the inference time complexity of the architectures does not change.

Another conclusive statement of this thesis is that architectures trained with different

[1] "Modified ResNet-50" name is coined by the author. This architecture is the one used in the non-local paper and has similar characteristics with R(2+1)D architecture. This architecture cannot be called as non-local architecture, since non-local is an attention concept and can be utilized with any architecture.

[2] R(2+1) Kinetics-400 pre-trained Top-1 performance is reported from another paper

modalities learn complementary information. In addition, this complementary information can be revealed even with a simple score fusion strategy. Considering the all of the 2D and 3D CNN experiments, two-stream (RGB + Flow) increases the Top-1 performance of the RGB stream from 6% to 10%, and the flow stream from 2% to 8 % in the HMDB51 dataset. Moreover, the utilization of RGB pose images created by the OpenPose algorithm as pose modality increases the Top-1 performance of 2D two-stream architectures by 3-4 % in the HMDB51 dataset, indicating that pose architectures are able to learn complementary information to RGB and Flow architectures.

For the utilization of non-local blocks, it is observed that utilizing non-local block does not always result in performance improvements. For instance, when we apply the original non-local block implementation as in the paper, the experiment result of 32x2 frame selection has been improved, but 64 frame selection becomes worse. In this thesis study, a non-local block is also utilized as an attention mechanism just before the late temporal modeling concept, such as concatenation or average pooling. It is observed that utilization with concatenation deteriorates the performance both in 2D and 3D CNN experiments. However, the utilization of non-local blocks with average pooling in 2D CNN has improved the performance.

To highlight, utilizing BERT as a late temporal modeling concept, utilization of late temporal modeling on 3D CNN architectures other than average pooling and the method of utilization of pose information are two novel parts of this thesis study. Moreover, this thesis fills some of the experiments gaps existing in the literature such as the experiments of SlowFast and modified ResNet on the HMDB51 dataset.

6.3 Future Work

A possible research direction might be proposals for parameter efficient BERT implementations that do not need feature reduction blocks (FRMB or FRAB) which decreases the capabilities of the final extracted features because of the reduction in the dimension of features.

One of the deficits of this thesis study is the lack of experiments of the proposed BERT-based late temporal modeling on a dataset which needs more temporal order

reasoning than HMDB51 and UCF101, such as something-something dataset.

The real benefits of BERT architecture rise to the surface with unsupervised techniques. For this reason, we have implemented proposed BERT-based distillation in order to benefit from unsupervised concepts. However, distillation is not a complete unsupervised training because the distilled architecture needs to be trained with supervised concepts. Therefore, as future work, better unsupervised concepts can still be proposed on BERT 3D CNN architectures.

Moreover, the experiments of the BERT-based distillation should be performed on a larger dataset than HMDB51, such as Kinetics because distillation is partially unsupervised techniques as argued in the previous paragraph and the benefits of the unsupervised techniques is proportional with the scale of the dataset.

REFERENCES

[1] U. Ahsan, R. Madhok, and I. Essa. Video jigsaw: Unsupervised learning of spatiotemporal context for video action recognition. In *Proceedings - 2019 IEEE Winter Conference on Applications of Computer Vision, WACV 2019*, pages 179–189. Institute of Electrical and Electronics Engineers Inc., 3 2019.

[2] Z. Cao, G. Hidalgo Martinez, T. Simon, S.-E. Wei, and Y. A. Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 7 2019.

[3] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman. A Short Note about Kinetics-600. 8 2018.

[4] J. Carreira, E. Noland, C. Hillier, and A. Zisserman. A Short Note on the Kinetics-700 Human Action Dataset. 7 2019.

[5] J. Carreira and A. Zisserman. Quo Vadis, action recognition? A new model and the kinetics dataset. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 4724–4733. Institute of Electrical and Electronics Engineers Inc., 11 2017.

[6] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng. Multi-fiber Networks for Video Recognition. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11205 LNCS, pages 364–380. Springer Verlag, 7 2018.

[7] V. Choutas, P. Weinzaepfel, J. Revaud, and C. Schmid. PoTion: Pose MoTion Representation for Action Recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 7024–7033. IEEE Computer Society, 12 2018.

[8] N. Crasto, P. Weinzaepfel, K. Alahari, and C. Schmid. MARS: Motion-augmented rgb stream for action recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, pages 7874–7883. IEEE Computer Society, 6 2019.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 10 2018.

[10] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *2005 IEEE International Workshop on Vi-*

sual Surveillance and Performance Evaluation of Tracking and Surveillance, pages 65–72, 10 2005.

[11] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell. Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691, 4 2017.

[12] W. Du, Y. Wang, and Y. Qiao. RPAN: An End-to-End Recurrent Pose-Attention Network for Action Recognition in Videos. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 3745–3754. Institute of Electrical and Electronics Engineers Inc., 12 2017.

[13] W. Du, Y. Wang, and Y. Qiao. Recurrent spatial-temporal attention network for action recognition in videos. *IEEE Transactions on Image Processing*, 27(3):1347–1360, 3 2018.

[14] L. Fan, W. Huang, C. Gan, S. Ermon, B. Gong, and J. Huang. End-to-End Learning of Motion Representation for Video Understanding. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 6016–6025. IEEE Computer Society, 12 2018.

[15] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2003.

[16] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-October, pages 6201–6210. Institute of Electrical and Electronics Engineers Inc., 10 2019.

[17] C. Feichtenhofer, A. Pinz, and A. Zisserman. 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, 12 2016.

[18] K. Gavrilyuk, R. Sanford, M. Javan, and C. G. M. Snoek. Actor-Transformers for Group Activity Recognition. pages 836–845, 3 2020.

[19] D. Ghadiyaram, M. Feiszli, D. Tran, X. Yan, H. Wang, and D. Mahajan. Large-scale weakly-supervised pre-training for video action recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:12038–12047, 5 2019.

[20] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman. Video Action Transformer Network. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:244–253, 12 2018.

[21] R. Girdhar and D. Ramanan. Attentional pooling for action recognition. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 34–45. Neural information processing systems foundation, 2017.

[22] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. ActionVLAD: Learning spatio-temporal aggregation for action classification. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.

[23] R. Goyal, S. E. Kahou, V. Michalski, J. Materzyńska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, F. Hoppe, C. Thurau, I. Bax, and R. Memisevic. The "something something" video database for learning and evaluating visual common sense. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:5843–5851, 6 2017.

[24] K. Hara, H. Kataoka, and Y. Satoh. Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet? In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 6546–6555. IEEE Computer Society, 12 2018.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 770–778. IEEE Computer Society, 12 2016.

[26] D. Hendrycks and K. Gimpel. Gaussian Error Linear Units (GELUs). 6 2016.

[27] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. 3 2015.

[28] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 4 2017.

[29] Ju Sun, Xiao Wu, Shuicheng Yan, L.-F. Cheong, T.-S. Chua, and Jintao Li. Hierarchical spatio-temporal context modeling for action recognition. pages 2004–2011. Institute of Electrical and Electronics Engineers (IEEE), 3 2010.

[30] M. E. Kalfaoglu, S. Kalkan, and A. A. Alatan. Late Temporal Modeling in 3D CNN Architectures with BERT for Action Recognition. In *European Conference on Computer Vision. Springer*, 2020.

[31] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. F. Li. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014.

[32] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The Kinetics Human Action Video Dataset. 5 2017.

[33] N. S. Keskar and R. Socher. Improving Generalization Performance by Switching from Adam to SGD. 12 2017.

[34] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 12 2015.

[35] A. Klaeser, M. Marszalek, and C. Schmid. A Spatio-Temporal Descriptor Based on 3D-Gradients. In *Procedings of the British Machine Vision Conference 2008*, 2008.

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

[37] I. Laptev. On Space-Time Interest Points. *International Journal of Computer Vision*, 64(2):107–123, 9 2005.

[38] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 6 2008.

[39] H. Y. Lee, J. B. Huang, M. Singh, and M. H. Yang. Unsupervised Representation Learning by Sorting Sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 667–676. Institute of Electrical and Electronics Engineers Inc., 12 2017.

[40] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. Snoek. VideoLSTM convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 166:41–50, 1 2018.

[41] J. Lin, C. Gan, and S. Han. TSM: Temporal Shift Module for Efficient Video Understanding. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-October:7082–7092, 11 2018.

[42] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik. Unifying distillation and privileged information. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 11 2015.

[43] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. *7th International Conference on Learning Representations, ICLR 2019*, 11 2017.

[44] W. Luo, Y. Li, R. Urtasun, and R. Zemel. Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, pages 4905–4913, 1 2017.

[45] P. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops 2009*, pages 514–521, 2009.

[46] R. Messing, C. Pal, and H. Kautz. Activity recognition using the velocity histories of tracked keypoints. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 104–111, 2009.

[47] J. Y. H. Ng, J. Choi, J. Neumann, and L. S. Davis. ActionFlowNet: Learning motion representation for action recognition. In *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*, volume 2018-Janua, pages 1616–1624. Institute of Electrical and Electronics Engineers Inc., 5 2018.

[48] J. Y. H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June, pages 4694–4702. IEEE Computer Society, 10 2015.

[49] A. Piergiovanni, A. Angelova, A. Toshev, and M. S. Ryoo. Evolving Space-Time Neural Architectures for Videos. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-October:1793–1802, 11 2018.

[50] A. Piergiovanni and M. S. Ryoo. Representation Flow for Action Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:9937–9945, 10 2018.

[51] D. Purwanto, R. Renanda Adhi Pramono, Y. T. Chen, and W. H. Fang. Extreme low resolution action recognition with spatial-temporal multi-head self-attention and knowledge distillation. In *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pages 961–969. Institute of Electrical and Electronics Engineers Inc., 10 2019.

[52] Z. Qiu, T. Yao, and T. Mei. 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., 12 2017.

[53] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245, 12 2013.

[54] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th international conference on Multimedia - MULTIMEDIA '07*, 2007.

[55] S. Sharma, R. Kiros, and R. Salakhutdinov. Action Recognition using Visual Attention. 11 2015.

[56] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, volume 2015-Janua, pages 802–810. Neural information processing systems foundation, 2015.

[57] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, volume 1, pages 568–576. Neural information processing systems foundation, 2014.

[58] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pages 4263–4270. AAAI press, 2017.

[59] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. 12 2012.

[60] C. Sun, F. Baradel, K. Murphy, and C. Schmid. Learning Video Representations using Contrastive Bidirectional Transformer. 6 2019.

[61] S. Sun, Z. Kuang, W. Ouyang, L. Sheng, and W. Zhang. Optical Flow Guided Feature: A Fast and Robust Motion Representation for Video Action Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1390–1399, 11 2017.

[62] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[63] D. Tran, H. Wang, M. Feiszli, and L. Torresani. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 5551–5560. Institute of Electrical and Electronics Engineers Inc., 4 2019.

[64] D. Tran, H. Wang, L. Torresani, J. Ray, Y. Lecun, and M. Paluri. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 6450–6459. IEEE Computer Society, 12 2018.

[65] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Å. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 5999–6009. Neural information processing systems foundation, 2017.

[66] H. Wang, A. Kläser, C. Schmid, and C. L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 2013.

[67] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

[68] L. Wang, P. Koniusz, and D. Q. Huynh. Hallucinating IDT Descriptors and I3D Optical Flow Features for Action Recognition with CNNs. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-October:8697–8707, 6 2019.

[69] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards Good Practices for Very Deep Two-Stream ConvNets. 7 2015.

[70] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016.

[71] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal Segment Networks for Action Recognition in Videos, 2018.

[72] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local Neural Networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 7794–7803. IEEE Computer Society, 12 2018.

[73] G. Willems, T. Tuytelaars, and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008.

[74] C. Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krahenbuhl. Compressed Video Action Recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 6026–6035. IEEE Computer Society, 12 2018.

[75] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 5987–5995. Institute of Electrical and Electronics Engineers Inc., 11 2017.

[76] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11219 LNCS, pages 318–335. Springer Verlag, 2018.

[77] D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie, and Y. Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, pages 10326–10335. IEEE Computer Society, 6 2019.

[78] A. Yan, Y. Wang, and Z. Li. PA3D : Pose-Action 3D Machine for Video Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7922–7931, 2019.

[79] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4713 LNCS, pages 214–223, 2007.

[80] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-Time Action Recognition with Enhanced Motion Vector CNNs. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 2718–2726. IEEE Computer Society, 12 2016.

[81] Y. Zhu, Z. Lan, S. Newsam, and A. G. Hauptmann. Hidden Two-Stream Convolutional Networks for Action Recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11363 LNCS:363–378, 4 2017.

APPENDIX A

OPTICAL FLOW

In this section, the brightness consistency equation in optical flow will be explained. Then the details about the TV-L1 algorithm will be given. TV-L1 is the most popular algorithm in the action classification literature. It depends on the total variation concept.

A.1 Brightness Consistency Equation

Brightness Consistency Equation depends on the two assumptions. The first one is the brightness consistency assumption which mainly depends on the idea that the moving pixel (projection of a point of the object to the camera plane) always has the same brightness. In a mathematical terminology, this can be given as $I\left(x\left(t\right),y\left(t\right),t\right)=C$. The second assumption is the small motion assumption so that the function can be approximated by first-order Taylor series expansion. For the small motion assumption, assume that the time interval between the frames is δt. Assume that the speed of the moving pixel is u and v for the x and y coordinates, respectively. Therefore, the pixel which is in the position of (x,y) in the previous frame, is in the position of $(x+u\delta t,y+v\delta t)$ in the next frame. The visualization of this small motion model can be seen in Figure A.1.

Then, from the brightness consistency approach, the pixel intensity will be the same for these frames such that

$$I(x,y,t)=I(x+u\delta t,y+v\delta t,t+\delta t). \tag{A.1}$$

Then, the function I can be linearized because the time interval δt is very small. The

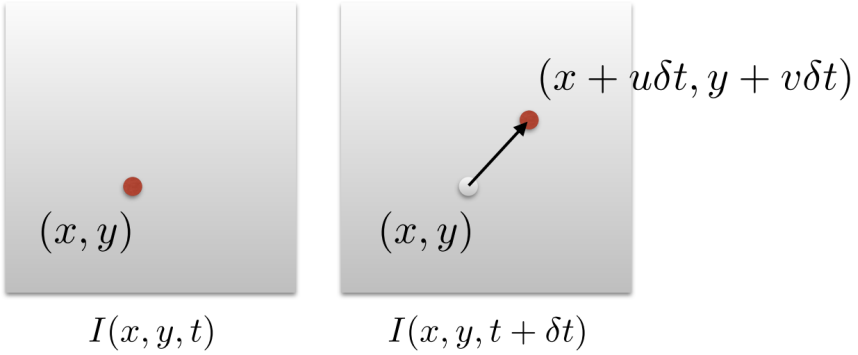


Figure A.1: Small motion model in a very small time interval Brightness Constancy, 16-385 Computer Vision (Kris Kitani), Carnegie Mellon University

multi-variate Taylor expansion for the function f is

$$f(x, y) = f(a, b) + f_x(a, b)(x - a) + f_y(a, b)(x - b). \tag{A.2}$$

Then, the Taylor series expansion of (A.1) is

$$I(x, y, t) = I(x+u\delta t, y+v\delta t, t+\delta t) = I(x, y, t)+\frac{\partial I(x, y, t)}{\partial x}\delta x+\frac{\partial I(x, y, t)}{\partial y}\delta y+\frac{\partial I(x, y, t)}{\partial t}\delta t. \tag{A.3}$$

The result of (A.3) is

$$\frac{\partial I(x, y, t)}{\partial x}\delta x + \frac{\partial I(x, y, t)}{\partial y}\delta y + \frac{\partial I(x, y, t)}{\partial t}\delta t = 0. \tag{A.4}$$

The division of (A.4) with δt yields

$$\frac{\partial I(x, y, t)}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I(x, y, t)}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I(x, y, t)}{\partial t} = 0. \tag{A.5}$$

The final form of these steps is

$$I_x u + I_y v = I_t, \tag{A.6}$$

which is called as brightness consistency equation and I_x, I_y and I_t are the image derivatives which can be calculated by any derivative filter like Sobel filter. u and v are the desired motion flow vectors.

A.2 TV-L1 Optical Flow Algorithm

In the previous section, the final form of brightness consistency equation is highlighted in (A.6). There are two unknowns in this equation, however there is only one

formula. Therefore, there is a need for extra constraint to solve this problem. This is known as aperture problem. Horn and Schunck solved this problem to some extent by adding L2 regularization term such that

$$\min_{u,v}\left(\int_{\Omega}\left(|\nabla u|^2+|\nabla v|^2\right)d\Omega+\lambda\int_{\Omega}\left(I\left(x+u\delta t,y+v\delta t,t+\delta t\right)-I\left(x,y,t\right)\right)^2 d\Omega\right), \tag{A.7}$$

where Ω denotes the number of pixels used for the optical flow estimation, u and v are horizontal and vertical parts of the motion vectors. In (A.7) the first term is regularization term and these second term optical flow term. The regularization term is also known as variational formulation. However, it is mentioned in [79] that penalizing deviations in a quadratic way does not allow discontinuities in motion. In this thesis, it is claimed that L_1 norm is much better than L_2 norm to handle such discontinuity problem. Denoting the intensities at time $t+\delta t$ as I_1 and at time t as I_0, the L1 version of (A.7) is

$$\min_{u,v}\left(\int_{\Omega}\left(|\nabla u|+|\nabla v|\right)d\Omega+\lambda\int_{\Omega}\left|I_1\left(x+u,y+v\right)-I_0\left(x,y\right)\right|d\Omega\right). \tag{A.8}$$

This optimization problem is tackled by adding a auxiliary variable (k,l), the new form of the function becomes

$$\int_{\Omega}\left(|\nabla u|+|\nabla v|\right)+\frac{1}{2\theta}\left((u-k)^2+(v-l)^2\right)+\lambda\left|\rho(k,l)\right|d\Omega. \tag{A.9}$$

Then, this is solved by two step iterative approach. One of the iteration to update u is

$$\min_{u}\int_{\Omega}\left(|\nabla u|\right)+\frac{1}{2\theta}(u-k)^2. \tag{A.10}$$

The other step is to update auxiliary variable k, which is

$$\min_{k}\int_{\Omega}\frac{1}{2\theta}(u-k)^2+\lambda\left|\rho(k,l)\right|. \tag{A.11}$$

This is to update u and k couple, but the procedure is the same with v and l couple. The further details about the solution this problem can be found in [79].

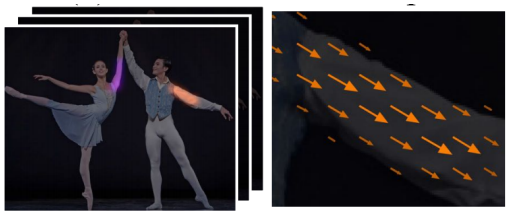
APPENDIX B

OPENPOSE

OpenPose [2] is a multi-person 2D pose estimator that utilizes deep architectures to estimate the positions of 25 predetermined joints. This technique is shown to perform quite promising during independent detection of the poses for every frame. OpenPose owes its popularity to the speed advantage compared to the other human pose extractor and is known as the first real-time multi-person 2D pose estimator.

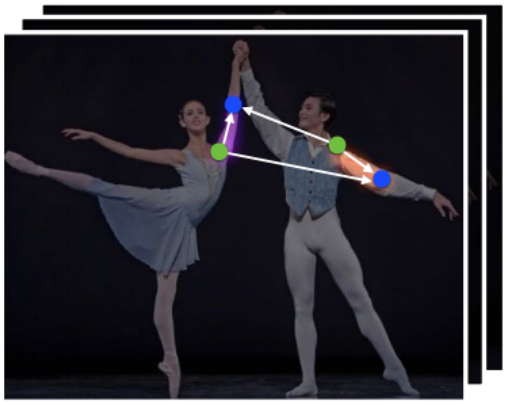
OpenPose algorithm consists of three main stages. These stages can be seen on the left side of Figure B.1. These stages are extraction of Part Affinity Fields (PAF), Part Confidence Maps (PCM), and Bipartite Matching. PAFs are to encode the part relations between the pre-defined joint relations (Limb). PAFs features preserve both the orientation and the positions. To make it clear, assume $j1$ and $j2$ are two joints. Then, define a unit vector that is oriented from $j1$ to $j2$. PAFs consists of the pixels which are located within the width of the limb. To encode the orientation information, two-channel is utilized. These channels encode the x and y components of limb orientation separately for the selected pixels. PCM is to denote the joints. The estimation of the joint locations is done by PCM. The ground truth joint locations are put as Gaussian locations to these maps. The pre-defined joints are shown in Figure B.1d. The number of PCM is equal to the number of pre-defined joints. Bipartite matching is related to the matching of the two joints according to a score calculated from the PAF.

To put it in a more clear way, assume that $\mathbf{L} = (\mathbf{L}_1, \mathbf{L}_2, ...\mathbf{L}_C)$ are the PAF, where C is the number of pre-defined limbs, and $\mathbf{S} = (\mathbf{S}_1, \mathbf{S}_2, ...\mathbf{S}_J)$ are PCM, where J is the number of joints. OpenPose has 25 joints and 26 limbs. $\mathbf{S}_j$ is a confidence map for a joint j and is a sample from $\mathbb{R}^{wxh}$. $\mathbf{L}_j$ is a part affinity field for a limb c and

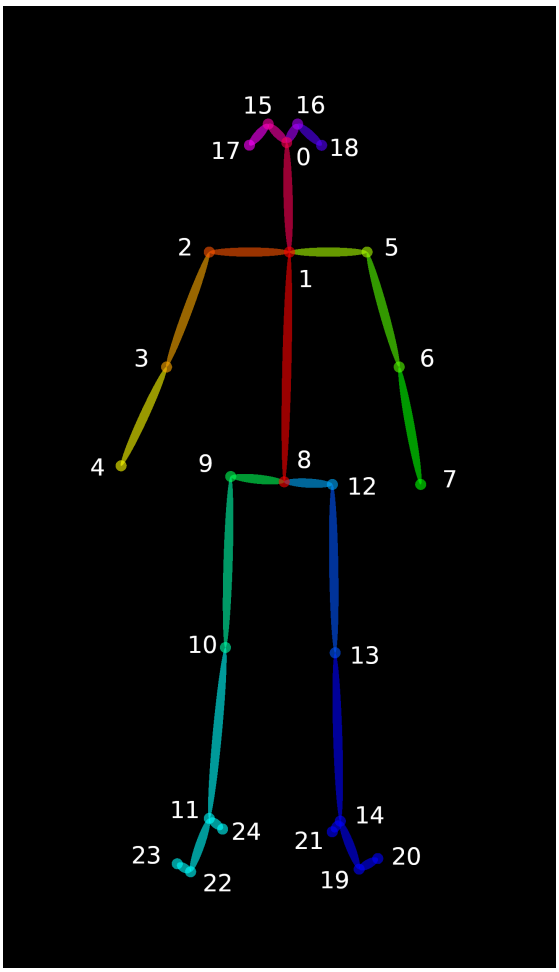


(a) Part Affinity Fields(PAF)

(b) Part Confidence Maps (PCM)



(c) Bipartite Matching



(d) Extracted Joints by OpenPose

Figure B.1: The Stages of the OpenPose algorithm (Left) and Extracted joints with OpenPose [2]

is a sample from. $\mathbb{R}^{wxhx2}$. Note that, PAFs are two dimensional vector for every pixel, while PCMs are scalar for every pixel. Assuming that $\mathbf{x}_{j,k}$ are the ground truth location for the j-th joint of the k-th person, $S^*_{j,k}$ can be defined as

$$\mathbf{S}^*_{j,k}(\mathbf{p}) = \exp \frac{||\mathbf{p} - \mathbf{x}_{j,k}||_2^2}{\sigma^2} \tag{B.1}$$

where $\mathbf{p}$ is the two dimensional location coordinate on PCM or PAF, and

$$\mathbf{L}^*_{j,k}(\mathbf{p}) = \mathbf{v} \text{ if } \mathbf{p} \text{ is on limb, } \mathbf{0} \text{ otherwise.} \tag{B.2}$$

where $\mathbf{v} = (\mathbf{x}_{j_1,k} - \mathbf{x}_{j_2,k})/||\mathbf{x}_{j_1,k} - \mathbf{x}_{j_2,k}||_2$. Being on the limb condition is defined as

$$0 \leq \mathbf{v} \cdot (\mathbf{p} - \mathbf{x}_{j_1,k}) \leq l_{c,k} \text{ and } \mathbf{v}_\perp \cdot (\mathbf{p} - \mathbf{x}_{j_1,k}) \leq \sigma_l \tag{B.3}$$

where $\mathbf{v}_\perp$ is the perpendicular vector vector of $\mathbf{v}$. $l_{c,k}$ denotes the length of the c-th limb of k-th person. σ_l is to denote the width of the limb.

For the details of the algorithm, the features of an image $\mathbf{I}$ is extracted by a function $f()$. This function is estimated by VGG-19 architecture in the paper. Then, PAFs of the I are tried to be estimated in an iterative manner by using the extracted features and previously estimated PAFs. Then, PCMs are tried to be estimated in an iterative manner by using the features, PAFs, and previously estimated PCMs. To explain it more clearly,

$$\begin{aligned}
\mathbf{F} &= f(\mathbf{I}) \\
\mathbf{L}^1 &= \phi^1(\mathbf{F}) \\
\mathbf{L}^t &= \phi^t(\mathbf{F}, \mathbf{L}^{t-1}) \quad \forall\, 2 \leq t \leq T_p \\
\mathbf{S}^{T_p} &= \rho^t(\mathbf{F}, \mathbf{L}^{T_p}) \\
\mathbf{S}^t &= \rho^t(\mathbf{F}, \mathbf{L}^{T_p}, \mathbf{S}^{t-1}) \quad \forall\, T_p \leq t \leq T_p + T_c
\end{aligned} \tag{B.4}$$

is the general form of the deep learning architecture, where ϕ is a PAF estimator and ρ is PCM estimator and T_p and T_c shows the number of iterations for PAF and PCM estimation, respectively.

The loss function for this iterative deep learning architecture is determined as

$$\begin{aligned}
\text{Loss}^{t_i}_{\mathbf{L}} &= \sum_{c=1}^{C} \sum_{\mathbf{p}} ||\mathbf{L}^{t_i}_c(\mathbf{p}) - \mathbf{L}^*_c(\mathbf{p})|| \\
\text{Loss}^{t_k}_{\mathbf{S}} &= \sum_{j=1}^{J} \sum_{\mathbf{p}} ||\mathbf{S}^{t_k}_j(\mathbf{p}) - \mathbf{S}^*_j(\mathbf{p})||
\end{aligned} \tag{B.5}$$

where $\mathbf{S}_j^*(\mathbf{p}) = \max_k \mathbf{S}_{j,k}^*(\mathbf{p})$ and $\mathbf{L}_c^*(\mathbf{p}) = \frac{1}{n_c(\mathbf{p})} \sum_k \mathbf{L}_{c,k}(\mathbf{p})$ and $n_c(\mathbf{p})$ denotes the number of people for the pixel coordinate $\mathbf{p}$.

In bipartite matching stage of OpenPose, the joints which belongs to the same person are aimed to be connected together. This is called multi-person parsing. For bipartite matching, firstly, there is a need to calculate the score (confidence) of association. For this score calculation, both PAF and PCM outputs are utilizes. The main idea behind this score metric is that the vector between the two joints in PCM has possibly the same direction encoding in PAF. The score function can be denoted as

$$Score = \sum_{\Phi} \mathbf{L}_c(\Phi) \frac{\hat{\mathbf{x}}_{j_1} - \hat{\mathbf{x}}_{j_2}}{||\hat{\mathbf{x}}_{j_1} - \hat{\mathbf{x}}_{j_2}||} \tag{B.6}$$

where $\hat{\mathbf{x}}_{j_1}$ and $\hat{\mathbf{x}}_{j_2}$ are the estimate of the locations of the joints j_1 andj_2, respectively and Φ is the set of pixels between the corresponding pixel locations of j_1 andj_2.

After calculating the confidence of the associations for every possible tuple (limb), the global solution can be obtained by maximizing the graph total score. However, in OpenPose the whole pose graph structure is subdivided into the multiple bipartite graphs. Therefore, every association is created independently from the other connected joints. This reduces the computation time during the inference of a pose. It is claimed that unconnected nodes are modeled by CNN architecture implicitly. To implement the multi-person parsing, firstly all connected parts are sorted following the score calculated in (B.6). Then every part is independently connected. When connecting the parts, if a person contradiction appears, such that the connected graph is already assigned to a different person, that part is ignored and not connected.

APPENDIX C

FURTHER DETAILS ABOUT EXPERIMENTS

C.1 Frame Selection Procedure for Late Temporal Modeling in 2D CNN Architectures

For temporal modeling, two main implementation types are applied in the literature. The first one is called the clip-based approach, for which clips consist of N number of frames without skipping any frames, such as [11]. During training, the clip is selected from a random position in a video sequence. During inference, multiple N-number of framed clips are extracted (with or without overlapping) from a video and the final decision is obtained as the mean average of the video clips. The main disadvantage of this scheme is that it has a high time complexity for the inference.

The other possible implementation is a video-based approach in which N number of frames are selected from the whole video [71, 13]. This selection can be with or without an equal interval between the frames. For the comparison of clip-based and video-based approaches, both of them are implemented by using LSTM. The clip-based approach yields 42.68% performance, while the video-based approach yields 47.12% performance on the split 1 of HMDB51. These results show that the utilization of the video-based approach is advantageous for both time complexity and performance.

During the training of the video-based approach, the video is divided into N number of segments, and a frame is selected randomly from each segment, resulting in frames without equal interval. During the inference, the same procedure is applied with a minor difference. Instead of selecting randomly, these frames are picked from the

middle of the segments. For both training and inference, multi-scale cropping is followed which is explained in Section 3.2.

C.2 Testing Procedures of 3D CNN Architectures

There are two important points during the test of any 3D CNN architecture. One of them is single-clip vs. multiple-clip settings. For the training of 3D CNN architectures, a clip is selected in a video at a random position. However, during the inference, it is possible to use multiple clips by averaging the results from every clip in a video. In multiple-clip settings, the clips are selected from a video in a non-overlapping manner. The number of clips in a video is equal to the floor of the division of the number of frames of a video to the number of frames of the clips. The other important setting is related to single-crop vs. ten-crops extraction from every clip. The information about ten crop test is presented in Section 3.2. Therefore, four possible test combinations are single clip - single crop, single clip - ten crops, multiple clips - single crop, and multiple clips - ten crops.

It should be emphasized that ten-crops might bring some performance improvements on the architectures but this technique comes with a ten-times computational increase.

C.3 Detailed Experimental Results for Different Clip and Crop Selections

The detailed view of the ResNeXt and I3D architectures on RGB, flow, and two-stream modalities is presented in Table C.1. In this table, the input size of the architectures, the number of frames in the clips, the input types of the architecture, and the architecture itself is shown in the column of the method. The other columns show the test types of architectures. The red color and blue color indicate the best Top-3 and Top-1 results of a specific method, respectively, among the four test types.

[1] Top1 result is denoted as 73.5 % in [8]. Three splits top1 average is denoted as 70.2 % in [24]

[2] Top1 result is denoted as 75.9 % in [8]

[3] Top1 result is denoted as 79.8 % in [8]

[4] Top1 result is denoted as 74.8 % in [49]. Three splits top1 average is denoted as 74.8 % and 74.3 % for Imagenet+Kinetics and Kinetics pre-trainings, respectively in [5]

[5] Top1 result is denoted as 77.1 % in [49]. Three splits top1 average is denoted as 77.1 % and 77.3 % for Imagenet+Kinetics and Kinetics pre-trainings, respectively in [5]

Table C.1: RESULTS OF TWO-STREAM ARCHITECTURES ON HMDB51 SPLIT-1 WITH FOUR TYPES OF TEST RESULTS

Method	Multiple Clips Ten Crops	Multiple Clips Single Crop	Single Clip Ten Crops	Single Clip Single Crop
ResNeXt101 RGB 112x112 64f [1]	Top3: 90.20 Top1: 73.07	Top3: 89.41 Top1: 73.73	Top3: 89.87 Top1: 72.88	Top3: 88.89 Top1: 73.20
ResNeXt101 Flow 112x112 64f [2]	Top3: 91.63 Top1: 79.80	Top3: 90.72 Top1: 79.74	Top3: 91.18 Top1: 78.50	Top3: 90.20 Top1: 78.50
ResNeXt101 Two-stream 112x112 64f [3]	Top3: 94.38 Top1: 82.35	Top3: 94.12 Top1: 81.83	Top3: 94.05 Top1: 81.96	Top3: 93.86 Top1: 81.24
I3D 224x224 64f [4]	Top3: 91.63 Top1: 74.90	Top3: 90.59 Top1: 74.64	Top3: 91.50 Top1: 74.51	Top3: 90.26 Top1: 74.51
I3D Flow 224x224 64f [5]	Top3: 91.18 Top1: 76.21	Top3: 91.50 Top1: 77.06	Top3: 90.98 Top1: 75.36	Top3: 91.24 Top1: 75.88
I3D Two-stream 224x224 64f [6]	Top3: 93.59 Top1: 80.59	Top3: 93.53 Top1: 80.46	Top3: 93.46 Top1: 80.59	Top3: 93.40 Top1: 80.20

C.4 FRMB implementation on 2D CNN Architectures with BERT-based late temporal modeling

An important detail about the usage of the BERT layer is the number of parameters of the BERT layer. This number is only 3M for ResNet18, while it reaches up to 50M for ResNet101, because of the difference in the dimension of the output sizes. 50M is a significantly larger and inefficient parameter size for any late pooling strategy since the ResNet101 backbone has a 45.20 M parameter which is less than the BERT layer.

Therefore, one possible remedy is to reduce the output size of the ResNet101 back-

[6] Top1 result is denoted as 80.1 % in [49]. Three splits top1 average is denoted as 80.7 % and 80.9 % for Imagenet+Kinetics and Kinetics pre-trainings, respectively in [5]

[7] Top1 result is denoted as 66.7 % in [8]. Three splits top1 average is denoted as 63.8 % in [24]

[8] Top1 result is denoted as 73.5 % in [8]. Three splits top1 average is denoted as 70.2 % in [24]

[9] Three splits top1 average is denoted as 75.4 % in [6]. The released weights of the authors also yields similar results with our reported results in our test scheme.

[10] Three splits top1 average is denoted as 77.4 % in [50]

[11] The selection procedure of frames in pre-training is as in [70]. Three splits top1 average is denoted as 73.5 % in [41]. The frame selection in fine-tuning and test seems ambiguous for me.

[12] Top1 result is denoted as 74.8 % in [49]. Three splits top1 average is denoted as 74.8 % and 74.3 % for Imagenet+Kinetics and Kinetics pre-trainings, respectively in [5]

[13] Top1 result is denoted as 80.1 % in [8]

[14] The pre-training is implemented with IG-65M, while the pre-training of other methods are implemented with Kinetics-400. Therefore, the dataset has also effect on obtaining the best performance in the table. Top1 result of R(2+1)D with Kinetics pre-training is denoted as 74.4 % in [49]. Therefore, about 7.7 % increase seems to be the result of the change in pre-training dataset.

bone. Such a decrease is possible by modifying the last block of ResNet101 or the addition of another block to the backbone just before the BERT layer. For this aim, *Feature Reduction with Modified Block* (FRMB) and *Feature Reduction with Additional Block* (FRAB) are proposed which are both explained in Section 4.1.2 for 3D architectures. The visual diagram for FRMB and FRAB is presented in Figure 4.2. This implementation reduces the number of parameters of the BERT layer from 50M to 3M again for the ResNet101 backbone. Moreover, FRMB reduces the number of parameters of the backbone as well.

Top-1 performances of flow BERT, pose BERT, two-stream and three-streams are 59.13%, 48.56%, 65.73%, and 68.62%, respectively, for ResNet101 with FRMB implementation. FRMB implementation increases the flow BERT and two-stream implementation about 1.5% and 0.5% amount, respectively; but decreases the pose BERT and three-stream implementation about 1% and 0.5%, respectively (See Table 3.7).

Table C.2: COMPARISON OF RECENT STATE OF THE ART ARCHITECTURES ON HMDB51 SPLIT-1 WITH FOUR TYPES OF TEST RESULTS

Method	Multiple Clips Ten Crops	Multiple Clips Single Crop	Single Clip Ten Crops	Single Clip Single Crop
ResNeXt101 112x112 16f [7]	Top3: 82.61 Top1: 63.33	Top3: 82.35 Top1: 62.09	Top3: 81.05 Top1: 62.35	Top3: 80.33 Top1: 60.65
ResNeXt101 112x112 64f [8]	Top3: 90.20 Top1: 73.07	Top3: 89.41 Top1: 73.73	Top3: 89.87 Top1: 72.88	Top3: 88.89 Top1: 73.20
MFNET 224x224 16f [9]	Top3: 87.58 Top1: 70.20	Top3: 87.52 Top1: 70.52	Top3: 86.34 Top1: 68.24	Top3: 88.82 Top1: 68.37
Rep-Flow-50 224x224 64f [10]	Top3: 89.54 Top1: 72.42	Top3: 89.02 Top1: 71.57	Top3: 88.04 Top1: 70.00	Top3:87.39 Top1:69.35
TSM 224x224 8f	Top3: 87.25 Top1: 66.80	Top3: 87.45 Top1: 67.97	Top3: 84.84 Top1: 64.12	Top3: 85.10 Top1: 64.71
TSM 224x224 8x8f	Top3: 89.54 Top1: 72.03	Top3: 89.15 Top1: 72.81	Top3: 89.35 Top1: 72.03	Top3: 89.02 Top1: 72.88
TSM [11] 224x224 8x8f	Top3: 90.13 Top1: 73.79	Top3: 89.28 Top1: 73.14	Top3: 89.87 Top1: 73.73	Top3: 89.15 Top1: 72.94
Modified ResNet50 224x224 32x2f	Top3: 88.69 Top1: 71.44	Top3: 88.76 Top1: 71.57	Top3: 88.69 Top1: 71.44	Top3: 88.56 Top1: 71.31
Modified ResNet50 Non-Local 224x224 32x2f	Top3: 91.44 Top1: 72.88	Top3: 90.33 Top1: 72.61	Top3: 91.31 Top1: 72.55	Top3: 90.26 Top1: 73.07
Modified ResNet50 224x224 64f	Top3: 91.70 Top1: 73.79	Top3: 90.92 Top1: 74.12	Top3: 91.44 Top1: 73.92	Top3: 90.78 Top1: 73.92
Modified ResNet50 Non-Local 224x224 64f	Top3: 90.26 Top1: 73.27	Top3: 89.54 Top1: 73.14	Top3: 89.87 Top1: 73.01	Top3: 89.22 Top1: 72.81
I3D 224x224 64f [12]	Top3: 91.63 Top1: 74.90	Top3: 90.59 Top1: 74.64	Top3: 91.50 Top1: 74.51	Top3: 90.26 Top1: 74.51
SlowFast-50 224x224 64f	Top3: 92.68 Top1: 78.37	Top3: 92.22 Top1: 78.69	Top3: 92.61 Top1: 77.65	Top3: 92.16 Top1: 78.43
MARS ResNext101 112x112 64f [13]	Top3: 92.75 Top1: 80.72	Top3: 92.03 Top1: 80.07	Top3: 92.35 Top1: 80.26	Top3: 91.44 Top1: 79.80
R(2+1)D ResNet34 [14] 112x112 32f	Top3: **93.86** Top1: 81.76	Top3: 93.46 Top1: **82.16**	Top3: 93.07 Top1: 81.83	Top3: 92.81 Top1: 82.22

Table C.3: THE RESULTS ON ALL SPLITS OF HMDB51 DATASET

	BERT	split-1	split-2	split-3	Average
R(2+1)D ResNet34 112x112 32f		Top3: 93.86 Top1: 81.76	Top3: 93.99 Top1: 82.03	Top3: 93.07 Top1: 79.87	Top3: 93.64 Top1: 81.22
R(2+1)D ResNet34 112x112 32f	✓	Top3: 95.16 Top1: 84.77	Top3: 95.23 Top1: 84.18	Top3: 94.71 Top1: 83.01	Top3: 95.03 Top1: 83.99
ResNeXt101 RGB 112x112 64f		Top3: 90.20 Top1: 73.07	Top3: 87.97 Top1: 73.46	Top3: 90.39 Top1: 76.14	Top3: 89.52 Top1: 74.22
ResNeXt101 RGB 112x112 64f	✓	Top3: 92.75 Top1: 77.25	Top3: 91.24 Top1: 77.52	Top3: 91.31 Top1: 77.71	Top3: 91.77 Top1: 77.49
ResNeXt101 Flow 112x112 64f		Top3: 91.63 Top1: 79.80	Top3: 90.65 Top1: 77.97	Top3: 91.70 Top1: 80.07	Top3: 91.32 Top1: 79.28
ResNeXt101 Flow 112x112 64f	✓	Top3: 92.88 Top1: 81.76	Top3: 92.29 Top1: 81.18	Top3: 92.42 Top1: 80.92	Top3: 92.53 Top1: 81.29
ResNeXt101 Two-Stream 112x112 64f		Top3: 94.38 Top1: 82.35	Top3: 92.16 Top1: 79.93	Top3: 93.27 Top1: 83.07	Top3: 93.27 Top1: 81.78
ResNeXt101 Two-Stream 112x112 64f	✓	Top3: 94.90 Top1: 83.99	Top3: 94.18 Top1: 83.46	Top3: 93.92 Top1: 83.20	Top3: 94.33 Top1: 83.55

Table C.4: THE RESULTS ON ALL SPLITS OF UCF101 DATASET

	BERT	split-1	split-2	split-3	Average
R(2+1)D ResNet34 112x112 32f		Top3: 99.26 Top1: 97.46	Top3: 99.68 Top1: 98.55	Top3: 99.84 Top1: 98.51	Top3: 99.59 Top1: 98.17
R(2+1)D ResNet34 112x112 32f	✓	Top3: 99.87 Top1: 98.63	Top3: 99.73 Top1: 98.90	Top3: 99.76 Top1: 98.43	Top3: 99.79 Top1: 98.65
ResNeXt101 RGB 112x112 64f		Top3: 98.55 Top1: 94.61	Top3: 98.77 Top1: 94.62	Top3: 99.03 Top1: 95.48	Top3: 98.78 Top1: 94.90
ResNeXt101 RGB 112x112 64f	✓	Top3: 98.89 Top1: 95.80	Top3: 99.09 Top1: 96.63	Top3: 99.32 Top1: 96.29	Top3: 99.1 Top1: 96.24
ResNeXt101 Flow 112x112 64f		Top3: 98.68 Top1: 95.56	Top3: 99.12 Top1: 95.96	Top3: 99.08 Top1: 96.35	Top3: 98.96 Top1: 95.95
ResNeXt101 Flow 112x112 64f	✓	Top3: 99.07 Top1: 95.74	Top3: 99.25 Top1: 96.92	Top3: 99.27 Top1: 96.83	Top3: 99.20 Top1: 96.49
ResNeXt101 Two-Stream 112x112 64f		Top3: 99.10 Top1: 97.20	Top3: 99.54 Top1: 97.38	Top3: 99.89 Top1: 97.81	Top3: 99.51 Top1: 97.46
ResNeXt101 Two-Stream 112x112 64f	✓	Top3: 99.60 Top1: 98.21	Top3: 99.29 Top1: 97.49	Top3: 99.89 Top1: 97.92	Top3: 99.59 Top1: 97.87

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Kalfaoglu, Muhammet Esat
Nationality: Turkish (TC)
Date and Place of Birth: 24.05.1994, Meram
Marital Status: Single
Phone: 0 555 6644177

EDUCATION

Degree	Institution	Year of Graduation
B.S.	Bogazici University	2017
Exchange	The University of Texas at Austin	2015

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2018 - 2020	OGAM	Project Personnel
2016	Aselsan	Intern

PUBLICATIONS

[1] M. E. Kalfaoglu, O. Can, B. Yildirim, and A. A. Alatan, "Antenna scan period estimation of radars with cepstrum and scoring," in *2019 27th Signal Processing and Communications Applications Conference (SIU)*, Apr. 2019, pp. 1-4. doi:

10.1109/SIU.2019.8806564.

[2] I. G. Dino, E. Kalfaoglu, A. E. Sari, S. Akin, O. K. Iseri, A. A. Alatan, S. Kalkan, and B. Erdogan, "Video content analysis-based detection of occupant presence for building energy modelling," in *CIB W78: Advances in ICT in Design, Construction and Management in Architecture, Engineering, Construction and Operations (AECO)*, 2019.

[3] I. G. Dino, A. E. Sari, E. Kalfaoglu, S. Akin, O. K. Iseri, A. A. Alatan, S. Kalkan, and B. Erdogan, "Automated building energy modeling for existing buildings using computer vision," in *CIB W78: Advances in ICT in Design, Construction and Management in Architecture, Engineering, Construction and Operations (AECO)*, 2019.

[4] F. C. Akyon and E. Kalfaoglu, "Instagram Fake and Automated Account Detection," *Proceedings - 2019 Innovations in Intelligent Systems and Applications Conference, ASYU 2019*, Sep. 2019. [Online]. Available: http://arxiv.org/abs/1910.03090.

[5] E. Kalfaoglu, I. G. Dino, O. K. Iseri, S. Akin, A. E. Sari, B. Erdogan, S. Kalkan, A. A. Alatan, Vision-Based Lighting State Detection and Curtain Openness Ratio Prediction in *Symposium on Simulation for Architecture and Urban Design (SimAUD)*, 2020

[6] E. Kalfaoglu, S. Kalkan, A. A. Alatan, Late Temporal Modeling in 3D CNN Architectures with BERT for Action Recognition, *16th European Conference on Computer Vision (ECCV), The 2nd Workshop on Video Turing Test: Toward Human-Level Video Story Understanding*, 2020