Hierarchical Action Classification with Network Pruning

by

Mahdi Davoodikakhki

B.Sc., Sharif University of Technology, 2018

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

© Mahdi Davoodikakhki 2020
SIMON FRASER UNIVERSITY
Summer 2020

Copyright in this work rests with the author. Please ensure that any reproduction or re-use is done in accordance with the relevant national copyright legislation.
Declaration of Committee

Name: Mahdi Davoodikakhki
Degree: Master of Science (Computing Science)
Thesis title: Hierarchical Action Classification with Network Pruning
Committee:

Chair: Yagiz Aksoy
Assistant Professor, Computing Science

Kang Yin
Supervisor
Associate Professor, Computing Science

Greg Mori
Committee Member
Professor, Computing Science

Yasutaka Furukawa
Examiner
Associate Professor, Computing Science
Abstract

Research on human action classification has made significant progresses in the past few years. Most deep learning methods focus on improving performance by adding more network components. We propose, however, to better utilize auxiliary mechanisms, including hierarchical classification, network pruning, and skeleton-based preprocessing, to boost the model robustness and performance. We test the effectiveness of our method on five commonly used testing datasets: NTU RGB+D 60, NTU RGB+D 120, Northwestern-UCLA Multiview Action 3D, UTD Multimodal Human Action Dataset, and Kinetics 400, which is a challenging and different dataset among the others. Our experiments show that our method can achieve either comparable or better performance on all the first four datasets. In particular, our method sets up a new baseline for NTU 120, the largest dataset among the first four. We also analyze our method with extensive comparisons and ablation studies.

Keywords: Human Action Recognition; Human Action Classification; Hierarchical Classification; Network Pruning
Dedication

To my parents and my wife for their inspiring support and help.
Acknowledgements

I would like to thank all the people, who helped me during my research and my Master’s studies. First, I would like to appreciate my senior supervisor, Professor KangKang Yin, who helped me a lot with her trust, concerns, and kindness. I would also like to thank the rest of the committee members, Professors Greg Mori, Yasutaka Furukawa, and Yagiz Aksoy for their help and suggestions.

In addition, I would like to thank all my lab mates in Graphics and Vision (GrUVi) lab at SFU especially Zeshi Yang, who helped me a lot in our research discussions and also provided the skeleton-based action classification method to be combined with our video-based action classification proposed methods. I also found my friends at SFU very helpful, particularly Mahta Khoshnam, Saeed Izadi, Mohammadreza Yavari, Zahra Mirikharaji, Amirhossein Safari, Mohammad Hadi Salari, and Hamed Shirzad, who helped me with their amazing ideas and supports.

I also found many other great friends in Vancouver. They helped me a lot in overcoming difficult situations and problems.
# Table of Contents

Declaration of Committee .......................................................... ii
Abstract ......................................................................................... iii
Dedication ...................................................................................... iv
Acknowledgements ........................................................................ v

Table of Contents ........................................................................... vi

List of Tables .................................................................................. viii

List of Figures ................................................................................ x

1 Introduction ............................................................................... 1
  1.1 Overview of Our Method .......................................................... 1
  1.2 Our Contributions ................................................................. 2
  1.3 Motivation ............................................................................. 2
  1.4 Thesis Organization ............................................................... 3

2 Related Work ............................................................................... 4
  2.1 Human Action Classification ................................................ 4
      2.1.1 Skeleton-based Action Classification ................................ 4
      2.1.2 Video-based Action Classification ................................... 5
  2.2 Hierarchical Classification and Loss Functions ...................... 6
  2.3 Network Pruning ................................................................... 7

3 Background .................................................................................. 8
  3.1 Datasets ................................................................................. 8
      3.1.1 Multi-modal Datasets ....................................................... 8
      3.1.2 Real-world Recorded Datasets ......................................... 8
  3.2 Base Network ......................................................................... 10
      3.2.1 ResNet Network ............................................................... 10
      3.2.2 Inflation Process .............................................................. 10
4 Method

4.1 Video Preprocessing ............................................. 13
4.2 Base Network ...................................................... 15
  4.2.1 Modified Inflated ResNet Architecture ...................... 15
  4.2.2 SlowFast Base Network ..................................... 15
4.3 Hierarchical Classification ..................................... 16
  4.3.1 Number of Superclass Sets ................................. 18
4.4 Network Pruning ................................................ 20
  4.4.1 Network Pruning Methods ................................. 20
  4.4.2 Network Pruning Importance .............................. 21
4.5 Other Auxiliary Methods ....................................... 21
4.6 Fusing Video-based and Skeleton-based Models .............. 21
  4.6.1 Skeleton-based Classification Method ..................... 22
  4.6.2 Fusion Methods ........................................... 22

5 Experiments ......................................................... 25

5.1 Implementation Details ....................................... 25
5.2 Network Architecture .......................................... 26
5.3 Hyperparameters .............................................. 26
5.4 Comparisons .................................................... 26
  5.4.1 Datasets .................................................. 27
  5.4.2 Performance .............................................. 30

6 Ablation Study ..................................................... 31

7 Conclusion .......................................................... 35
  7.1 Limitations ................................................... 35
  7.2 Future Work .................................................. 35

Bibliography ........................................................ 37

Appendix A Convolutional Layers Weights ......................... 42
### List of Tables

| Table | Description | Page |
|-------|-------------|------|
| Table 3.1 | Number of blocks in each ResNet version. | 11 |
| Table 4.1 | Comparison on Kinetics 400 using video-based classification. | 15 |
| Table 4.2 | Different weighting schemes of the hierarchical loss function on the NTU 60 and Kinetics 400 datasets. | 19 |
| Table 4.3 | Comparing our network model with network pruning and dropout on the NTU RGB+D 60 dataset. Here the full model refers to the inflated ResNet50 trained with the hierarchical loss using cropped video input. | 20 |
| Table 4.4 | Comparing reinitializing weights with inherited weights for iterative pruning on the NTU RGB+D 60 dataset. The full model refers to the baseline network trained with the hierarchical loss using cropped video input. | 20 |
| Table 4.5 | Different pruning ratios and passes on the NTU 60 cross-subject benchmark. | 21 |
| Table 4.6 | Fusion results on NTU 120 and Kinetics 400. The first two methods are the baseline models, and the next three are fusion results by weighted averaging. The last row is classification using concatenated features from both models with an MLP. | 22 |
| Table 5.1 | Different Inflated ResNet Architectures on NTU 60. | 26 |
| Table 5.2 | Different superclass sizes for level 1-3 classification on the NTU 60 dataset. | 27 |
| Table 5.3 | Different superclass sizes for level 1-3 classification on the Kinetics 400 dataset. | 27 |
| Table 5.4 | Comparison on NTU 60. – indicates no results available. | 27 |
| Table 5.5 | Comparison on NTU 120. * indicates results obtained from author-released code. – indicates no results available. Our fusion results are obtained by fusing the skeleton-based and the video-based action classification models. | 28 |
| Table 5.6 | Comparison on N-UCLA. – indicates no results available. The Pre-trained column indicates if the model was pre-trained on ImageNet and/or a bigger human action dataset. | 29 |
### Table 5.7 Comparison on UTD-MHAD
* indicates results obtained from author-released code. The Pre-trained column indicates if the model was pre-trained on ImageNet and/or a bigger human action dataset.

### Table 5.8 Comparison on Kinetics 400 using video-based classification
* indicates results obtained from author-released code. – indicates no results available. Our fusion results are obtained by fusing the skeleton-based and the Inflated ResNet50 video-based action classification models.

### Table 6.1 Ablation Study on NTU 60 dataset
The baseline model refers to the Inflated ResNet50 network trained without hierarchical loss using the original videos. The full model refers to the baseline network trained with the hierarchical loss using cropped video input.

### Table 6.2 Ablation Study on NTU 120 dataset

### Table 6.3 Ablation Study on N-UCLA dataset
The “Viewn” columns indicate the camera view used for testing and the other two views are used for training.

### Table 6.4 Ablation Study on UTD-MHAD dataset

### Table 6.5 Ablation Study on Kinetics 400 dataset with our methods and Inflated ResNet50 base network
Here, full model does not contain the video crop preprocessing stage.

### Table 6.6 Ablation Study on Kinetics 400 dataset with our methods and Slow-Fast base network
Here, full model does not contain the video crop preprocessing stage.

### Table A.1 Inflated ResNet50 convolutional and total network weights
List of Figures

Figure 3.1 Examples from action classification datasets. The action video frames in each row are from one video in each dataset as follows. a) Sitting down action from NTU RGB+D 60 dataset b) Shake fist action from NTU RGB+D 120 dataset c) Jog action from UTD-MHAD dataset d) Donning action from N-UCLA dataset e) Eating watermelon from Kinetics 400 dataset f) Flipping pancake from Kinetics 400 dataset.

Figure 3.2 Examples of NTU dataset data types for a video from "shaking hands' action class.

Figure 3.3 Examples of Kinetics dataset RGB frames and RGB frames with 2D skeleton for a sample video from "cartwheel" action class.

Figure 3.4 Example frames from Kinetics 400 dataset RGB frames for a video from "Cooking chicken" action class.

Figure 3.5 ResNet building blocks.

Figure 3.6 Original and Inflated ResNet BottleNeck building blocks.

Figure 4.1 Projection of a 3D skeleton onto the image plane as a 2D skeleton.

Figure 4.2 Structure of our neural network.

Figure 4.3 NTU 60: All 60 action classes and the derived superclasses.

Figure 4.4 N-UCLA: All 10 action classes and the derived superclasses.

Figure 4.5 Averaged and normalized absolute gradients in the last convolutional layer of stack 1, backpropagated from different levels of classifiers. The gradients are computed from the first 500 clips of the NTU 60 dataset. Pure red indicates 1 and pure blue indicates 0.

Figure 4.6 Example video frames and Skeleton data. The first rows of videos and Skeleton data are from NTU RGB+D 60 dataset, and the second rows are from Kinetics 400 dataset. Fusion can be a simple weighted averaging or an MLP.

Figure 6.1 Example frames from UTD-MHAD dataset. Each row contains frames from a single video and their actions are as follows. a) right arm swipe to the right b) right hand draw x c) bowling (right hand) d) two hand push e) stand to sit.
Chapter 1

Introduction

Human action classification and recognition has many important applications, such as autonomous driving, smart surveillance, patient monitoring, and interactive and performance games. Despite extensive research on this topic in recent years, human-level performance is still out of reach. Image classification; however, has achieved human-level performance a few years ago. There are many challenges in human action classification and recognition. First, there are high intra-class variations and inter-class similarities. A powerful deep learning model and a large amount of training data are necessary to achieve good performance. Second, the qualities of input videos vary greatly. There are multiple benchmark datasets, and in this work, we focus on multi-modal dataset videos captured in indoor lab environments, however, we still test the efficiency of our methods on Kinetics400, which contains both indoor and outdoor videos. Third, multiple data types and representations can be captured with the video data or extracted from the videos. Skeleton data, for example, should be used whenever possible. Last but not least, because most of the video-based methods use several RGB frames as their input, they are computationally more complex than image-based methods. Consequently, the training phase is more time consuming, and finding the optimal hyper-parameters and configuring the models are more challenging.

1.1 Overview of Our Method

Considering the similarities between image classification and video action classification, our work in video action classification is inspired by relevant researches in image classification area. We propose to extend the Inflated ResNet architecture with hierarchical classification for better feature learning at different scales. Iterative pruning is then incorporated for a further performance boost. We also use skeleton data, captured or extracted, to crop out irrelevant background, so the learning can focus on human activities. These mechanisms combined set up a new baseline for the newly-released large-scale dataset NTU RGB+D 120. Finally, we fuse our proposed model with a skeleton-based action classification method
from our research group to take advantage of both video-based and skeleton-based action classification methods.

1.2 Our Contributions

In summary, our main contributions include:

- We show that Inflated ResNet coupled with hierarchical classification can boost the performance of the baseline model.

- We show that iterative pruning can help improve the performance even further. We also investigate two different methods of pruning convolutional layers and compare their performance.

- We also show that 2D/3D skeleton data, when available, could be used to crop videos in a preprocessing stage to increase the classification accuracy in most cases. Besides, to the best of our knowledge, we are the first to use the skeleton data for video cropping in the preprocessing stage.

- We evaluate our method extensively on five datasets. Our method sets up a new baseline for the NTU RGB+D 120 dataset for future research in this field.

- We evaluate our method on Kinetics 400, which is larger than the other experimented datasets and is more similar to the real-world videos because it is collected from YouTube videos captured and uploaded by this social media users. These attributes make this dataset the most challenging one among the other datasets.

- To boost the performance even more, we fuse our final model with a skeleton-based action classification method proposed by our research group to boost the performance even further for the two largest experimented datasets, namely NTU RGB+D 120 and Kinetics 400.

1.3 Motivation

Many research studies in human action classification area focus on proposing novel and relatively complex methods to boost the state-of-the-art performance but overlook the learning procedure and training issues such as over-parameterization and redundancy in convolutional layers. It is also worth noting that many of these methods share the same or very similar backbone networks such as different variations of ResNet networks. The methods presented in current study are designed for improving the generalization of the learned parameters and addressing the over-parameterization in convolutional layers. Moreover, They can be implemented and utilized to improve most of the state-of-the-art researches and possibly future researches in this area.
1.4 Thesis Organization

The rest of this thesis is organized as follows; in Chapter 2, we review related studies related to human action classification and our proposed methods. We provide some backgrounds about the experimented datasets and our base network in Chapter 3. Chapter 4 contains a detailed description of our proposed methods. Chapter 5 is dedicated to implementation details and our experiments on five common action classification datasets. Chapter 6 covers our ablation studies. Finally, Chapter 7 concludes this work while discussing its limitations and suggesting possible directions for further research.
Chapter 2

Related Work

Due to the increasing interest in human action classification over the past years, there is a large body of prior studies related to our work. In this section, we summarize the most recent relevant papers.

2.1 Human Action Classification

Most human action classification methods work on either RGB image sequences and/or skeleton data. Our method uses both 2D skeletons and video inputs for classification, so we will review and compare with state-of-the-art methods from both categories.

2.1.1 Skeleton-based Action Classification

For skeleton-based classification, traditional CNN (Convolutional Neural Networks) methods can still be used after converting skeleton data into 2D images. Example works include TSRJI [3], Skelemotion [4], Enhanced Viz. [34], JTM [51], JDM [25], and Optical Spectra [19]. RNN (Recurrent Neural Networks) and its two common variations LSTM (Long Short-Term Memories) and GRU (Gated Recurrent Units) can also be used to interpret skeleton sequences. Their ability to learn long and short-term memories help achieve good results. Example works include TS-LSTM [24] and EleAtt-GRU [60].

To improve the performance, a variety of techniques have been proposed. The most dominant ones are: splitting joints into five groups [41], [49], [59], using attention modules to focus on important joints and frames [29], [60], transforming joint coordinates into body local frame or other frames [24], [41], calculating joint movements between consecutive frames [24], and traversing body joints in a specific order [31], [49]. Recently, GCN networks have been popular with their superior performance, which is partly due to their ability for modeling the sparse adjacency among human joints [27, 28, 36, 37, 39, 42, 44, 45, 47, 55, 56, 57].
2.1.2 Video-based Action Classification

Experiments on Multi-model Datasets

Most state-of-the-art video-based classification methods are based on CNNs. Inflated 3D ConvNet (I3D) and Inflated ResNet proposed by [8] became the foundation of many advanced algorithms, such as MMTM [21], Glimpse Clouds [2], Action Machine [62], and PGCN [43]. Such networks inflate 2D kernels of convolutional filters and pooling layers along the temporal domain to process 3D spatio-temporal information. In addition, Glimpse Clouds [2] extracts attention glimpses from each frame and uses the penultimate feature maps to estimate 2D joint positions and encourage glimpses to focus on the people in the scene. Action Machine [62] extracts Region of Interests (RoI), which are human bounding boxes, for better pose estimation and classification over these regions. The skeleton classification results are then fused with video-based classification to boost the performance further. PGCN [43] performs graph convolutions over RGB features extracted around 2D joints rather than over the joint positions to improve the video-based classification performance, which is then also fused with skeleton-based classification scores. Our baseline network is similar to that of Glimpse Clouds [2], and our cropping preprocess is inspired by Action Machine [62].

In addition to video and skeleton input, various other data types can be used for input or intermediate feature representations. For instance, PoseMap [33] extracts pose estimation maps from the RGB frames. 3D optical flow can also be estimated for classification [1]. RGB and optical flow classifications can be fused to further boost performance [46]. MMTM [21] uses a combination of depth, optical flow, and skeleton data with RGB frames as input for different datasets. Different fusion strategies have also been investigated, such as MMTM [21] and [40] that fuse features from intermediate layers for the next layers or final classification.

Experiments on Real-world Recorded datasets

Instead of focusing on indoor datasets such as NTU RGB+D dataset [30, 41], some methods consider more challenging and low-quality datasets, such as those created by collecting videos captured by ordinary people, e.g., Kinetics 400 [23], Kinetics 600 [6] and Kinetics 700 [7] and AVA-Kinetics datasets [16]. Methods working on this type of datasets typically design deeper models or use larger base networks to better overcome faced challenges. The majority of these state-of-the-art methods are supported by giant technology companies, such as Facebook [12, 15, 52, 53] and Amazon [11], and most of their experiments are run on powerful machines with 128 GPUs [12, 15, 53], which makes performing similar researches on these datasets harder for smaller research groups. Here, we focus on the most recent and related researches performed on the Kinetics 400, as we have also performed some experiments on this dataset.
Wang et al. [52] proposed non-local blocks made of non-local operations to capture long-range dependencies. In this work, they addressed the constraint of the convolutional and recurrent networks in capturing local neighborhood information by considering the relations between different and possibly distant features in the space-time domain. Moreover, with the residual connections designed in their non-local blocks, their method can be added to the existing proposed networks to boost the performance of these networks. A model with two pathways was introduced in [12]. The first pathway, called the slow path, is the larger of the two and is designed for capturing spatial features. The second pathway, namely the fast path, perceives temporal features and receives video frames at a higher frame rate. As this pathway receives more frames, it is designed to be smaller, to keep the computational complexity in balance between the two paths. A multigrid training, which is a scheduled training with varied mini-batches and different input dimension sizes along the time and space to improve both performance and training speed, was also proposed in [53].

Another set of methods collected large and similar datasets to the original Kinetics dataset from social media, such as Instagram, and could achieve a high performance with extensive pre-training. Such methods are mostly based on previously developed techniques and do not necessarily seek a novel solution to improve the performance in video-based action classification. Ghadiyaram et al. [15] collected a large pre-training set with more than 65 million videos with multiple labels for each of them to pre-train their network and found their pre-training to be beneficial even considering the noises that occur when collecting the data with user-provided hashtags. Duan et al. [11] addressed the problem of the noise and the large size of the pre-training dataset in [15], which was computationally expensive during the pre-training stage, by using a pre-trained teacher model on the Kinetics 400 dataset to filter out the irrelevant and noisy videos. Moreover, they proposed a unique format for images and trimmed and untrimmed videos to be used in the pre-training. They were able to improve the previous work with a significantly smaller but more curated collected dataset with approximately 3.5 million images and 800 thousand of videos.

2.2 Hierarchical Classification and Loss Functions

Hierarchical classification and loss functions facilitate learning the most important features in different scales. One straightforward way to apply hierarchical classification is to learn on different resolutions of the same set of images, such as [22] for skin lesion classification. Semantic graphs can be constructed to form hierarchical relations among classes for text classification [54]. Our approach is mainly inspired by related works in image classification that use features from intermediate layers for the main classification as well, by accumulating loss functions from all participating layers [20, 48].
2.3 Network Pruning

Over-parameterization is a well-known property of deep neural networks. Network pruning is usually used to improve generalization and achieve more compact models for low-resource applications [14, 35]. There are multiple choices to implement pruning and the fine-tuning after pruning. One option is to use one-shot pruning [26], but usually unimportant filters in convolutional layers are iteratively located and deleted. After each pruning iteration, the network can be re-trained from scratch with reinitialized weights [14, 35]. Our pruning method is similar to [14], as we also iteratively prune filters with the lowest $l_2$ norms. The difference is that we retrain with inherited weights, similar to [26].
Chapter 3

Background

3.1 Datasets

We categorize the human action classification classes into two main groups, namely, multi-modal datasets and real-world recorded datasets. Figure 3.1 shows a few examples from the experimented datasets.

3.1.1 Multi-modal Datasets

The main human action classification datasets we focus on performing experiments are Multi-modal datasets. We have performed experiments on NTU RGB+D 60 Dataset [41], NTU RGB+D 120 Dataset [30], Northwestern-UCLA Multiview Action 3D (N-UCLA) [50], and UTD Multimodal Human Action Dataset (UTD-MHAD) [9], which are in this category of datasets and all of them are captured with Microsoft Kinect sensor.

These datasets are mostly recorded in in-door environments and have multiple types of inputs such as Depth frames, Video RGB frames, and 3D and possibly 2D skeletons. Several of these datasets provide more data such as IR sequences, depth sequences, and depth-masked frames in NTU RGB+D 60 and 120, or wearable inertial sensor data in UTD Dataset. Figure 3.2 illustrates examples of several NTU dataset data types.

3.1.2 Real-world Recorded Datasets

There is another set of datasets, which are collected in the real-world situations and environments. Kinetics 400 [23], Kinetics 600 [6], and Kinetics 700 [7] are among such datasets. Since these dataset videos are captured from the videos uploaded by general users in social media, such as YouTube, they are very different from one another in terms of quality, Frames Per Second (FPS), environment, aspect ratio, subjects, and semantics.

The mentioned differences make these datasets more challenging. For these datasets, we only have the RGB frames, and several methods, such as [56], have used OpenPose [5] for extracting 2D skeletons from the provided videos. Figure 3.3 shows the original frames as well as the frames with the extracted 2D skeletons drawn over them [56]. However,
Figure 3.1: Examples from action classification datasets. The action video frames in each row are from one video in each dataset as follows. a) Sitting down action from NTU RGB+D 60 dataset b) Shake fist action from NTU RGB+D 120 dataset c) Jog action from UTD-MHAD dataset d) Donning action from N-UCLA dataset e) Eating watermelon from Kinetics 400 dataset f) Flipping pancake from Kinetics 400 dataset.

Figure 3.2: Examples of NTU dataset data types for a video from "shaking hands" action class.

several videos, people do not exist or are barely visible. Therefore, the skeleton-based action classification and video cropping around people becomes very challenging. Figure 3.4 shows an example of such videos.
3.2 Base Network

Our proposed networks are based on ResNet, which was primarily designed for image classification in ImageNet challenge and shown to be efficient. The base ResNet network was further enhanced by inflating the convolutional and pooling kernels along the temporal dimension to make it suitable for extracting features from videos, which is more discussed in 3.2.2.

3.2.1 ResNet Network

Here, we discuss the details of ResNet network, which constructs the base network architecture for our methods. ResNet was first introduced in [17] and has five main variations, all of which have 5 main layers, namely ResNet stacks. All stacks except the first one are constructed by building blocks. More specifically, ResNet18 and ResNet34 are composed by building blocks with 2 convolutional layers, while the other three, i.e., ResNet50, ResNet101, and ResNet152, are composed by bottleneck building blocks with three convolutional layers. Figure 3.5 shows a diagram of the two different building blocks in ResNet and Table 3.1 shows the number of blocks in each stack.

3.2.2 Inflation Process

In the past few years, many successful and famous methods have been proposed for image classification and image recognition. To take advantage of such networks, methods such as [8] and [13] have tried to inflate the 2D convolutional and pooling kernels to 3D ones along the temporal domain to make these modified networks suitable for video inputs. In the first inflation method used in this work from [8], the square 2D kernels in the ResNet
networks are inflated into 3D kernels and are created by repeating the third dimension equal to the first two ones to make a cubic kernel. Such an inflation technique is used in our first method, which we call the Inflated ResNet50 base network, used previously in [2]. The second method used for inflation is proposed by [13], in which the temporal dimension of the first convolutional layers in the 3D BottleNeck building blocks is equal to 3, while for the other convolutional layers, the temporal dimension is 1. This inflation method is used in the SlowFast [12] network, which we use as the base network to test our methods on the Kinetics 400 dataset. Figure 3.6 shows the original and the two inflated ResNet bottleneck building blocks.
Figure 3.6: Original and Inflated ResNet BottleNeck building blocks.
Chapter 4
Method

In this chapter, we present our video preprocessing procedures including cropping and projecting 3D skeletons to 2D in Section 4.1. We describe our modified ResNet network architecture in Section 4.2. We then detail the hierarchical classification and network pruning in Sections 4.3 and 4.4. We also briefly describe the auxiliary methods we used for training in Section 4.5. Finally, having developed our final video-based action classification model, we describe the methods used for fusing our model with a skeleton-based action classification model in Section 4.6. Also, we would like to note that the majority of our proposed methods and experiments are also available in [10].

4.1 Video Preprocessing

Raw action videos usually contain not only human subjects, but also surrounding objects and background environments, most of which are irrelevant to the performed actions. Neural networks can overfit to such noises instead of focusing on human actions. Inspired by [62], we also crop the multi-modal datasets’ raw videos to focus on humans present inside. Video cropping preprocessing alone is beneficial in improving performance for three out of five experimented datasets as shown in our ablation studies in Chapter 6 in Tables 6.1-6.3.

We use 2D skeletons and joint positions in pixels for cropping. 2D skeleton data can be captured together with the videos, or extracted by pose estimation algorithms such as OpenPose [5], or computed from 3D skeletons as we will explain shortly. We first extract the skeleton bounding boxes, and then enlarge them by a factor of 10% on all four sides and cap them at frame borders in order to leave a margin for errors and retain relevant information of surrounding areas. After cropping, we rescale all the video frames to a resolution of $256 \times 256$ as input to our neural network.

For datasets that only provide 3D skeleton data, we project the 3D skeletons onto the image plane as 2D skeletons as illustrated in Figure 4.1 following Equation 4.1. We denote the 2D and 3D skeletons as $S_p \in R^{T \times J \times 2}$ and $S \in R^{T \times J \times 3}$, respectively, where $T$ is the number of frames and $J$ is the number of joints. We denote the individual channels in
skeleton data as $S_{px}, S_{py}$ for 2D pixel positions, and $S_x, S_y, S_z$ for 3D world coordinates. $b_x = 320$ and $b_y = 240$ are bias values that correspond to the image centers for both the N-UCLA and UTD-MHAD datasets. $c_x, c_y$ are coefficients that can be solved for from Equation 4.2 in a semi-automatic approach. We randomly sample ten frames from UTD-MHAD video clips from ten different action classes and manually estimate the pixel position $S_{px}$ and $S_{py}$ of 5 end-effector joints (head, hands, and feet). A least squares fit returns $c_x = 558.1$ and $c_y = 579.5$. These coefficients work well for both the N-UCLA and UTD-MHAD datasets.

$$S_p = \begin{bmatrix} S_{px} \\ S_{py} \end{bmatrix} = \begin{bmatrix} c_x \times \frac{S_x}{S_z} + b_x \\ c_y \times \frac{S_y}{S_z} + b_y \end{bmatrix}$$ (4.1)

$$\begin{bmatrix} c_x \\ c_y \end{bmatrix} = \begin{bmatrix} (S_{px} - b_x) \times \frac{S_x}{S_z} \\ (S_{py} - b_y) \times \frac{S_y}{S_z} \end{bmatrix}$$ (4.2)

As mentioned earlier, video cropping alone is beneficial for three out of five experimented datasets, which are NTU RGB+D 120, NTU RGB+D 60, and N-UCLA datasets as shown in Tables 6.1-6.3. It also improves the performance when combined with hierarchical classification for the UTD-MHAD dataset as shown in Table 6.4. We do not use cropping for Kinetics 400 because we did not find it beneficial with our Inflated ResNet50 base network in our early experiments (see Table 4.1). The main reason could be that in many video frames people are barely visible or do not exist as shown in the example in Figure 3.4 and discussed.
Table 4.1: Comparison on Kinetics 400 using video-based classification.

| Method                  | Top-1  | Top-5  |
|-------------------------|--------|--------|
| ours with cropping      | 63.40% | 84.72% |
| ours without cropping   | 64.51% | 85.28% |

earlier in [43]. The early experiments were obtained using the recently downloaded Kinetics 400 dataset. This dataset is smaller, and the format of its videos is slightly different with that of videos in the dataset used in a previous work [52]. In our smaller dataset, videos were resized to the fixed size of $256 \times 256$, but in the dataset used in [52], videos were resized to have a height of 256 while keeping the aspect ratio.

4.2 Base Network

We use two different base networks to determine the efficiency of our methods in human action classification, which we briefly explain in this section. We use Inflated ResNet50 architecture from [2] for all of the five experimented datasets and use SlowFast [12] as base network particularly for the Kinetics 400.

4.2.1 Modified Inflated ResNet Architecture

Our baseline network is the Inflated ResNet from Glimpse Clouds [2], which was created according to the inflation procedure introduced in [8]. The Inflated ResNet is a variation of ResNet developed by [17]. It is also similar to the I3D network in [62]. In Inflated ResNet, 2D convolutional kernels, except the first one, are converted into 3D kernels, to make them suitable for video input. We perform experiments on different variations of Inflated ResNet in Section 5.2, and find the Inflated ResNet50 to be the best architecture for our task.

Our baseline network consists of four main ResNet stacks of convolutional layers, each consisting of multiple bottleneck building blocks. We label them as stacks 1 to 4 in Figure 4.2 for hierarchical classification. More specifically, we modify the baseline network after each ResNet stack by averaging the extracted features over the spatial and temporal domain and then passing them to a fully-connected linear layer and a softmax layer to obtain multiple levels of classification probabilities.

4.2.2 SlowFast Base Network

The Inflated ResNet50 base network with our methods performs very well and is comparable with the state-of-the-art methods on multi-modal and indoor datasets. However, it does not perform as great as the other state-of-the-art methods on Kinetics 400 dataset. Therefore, we also implement and test our methods on top of the SlowFast [12] network, which contains two slow and fast paths, based on ResNet50 to test them on this new and more powerful
network as well. Besides the SlowFast base network, we use the multigrid training [53], which was added to the SlowFast by the original network creators, to test our methods and perform the training faster. We utilize the same architecture for the hierarchical classification used in Inflated ResNet50 for the SlowFast base network. We concatenate the features from both paths after each stack and pass them to the same structure of fully-connected layers having more input features than the previous network. We increase the fully-connected layers’ input size to make them suitable for capturing the concatenated features from both of the slow and fast paths.

### 4.3 Hierarchical Classification

We employ hierarchical classification to encourage our neural network to learn important features at different scales. For each ResNet stack, we enforce a superclass constraint for each action. That is, a fully-connected linear classifier assigns each action to a superclass after each ResNet stack. Each superclass contains the same number of original action classes to keep the learned hierarchy balanced. We use up to four ResNet stacks for hierarchical classification. The final structure of our network is shown in Figure 4.2. We train the classifiers in two passes. The first pass trains the network with a cross-entropy loss only after the last stack. We then compute the confusion matrix $C$ from the trained network for assigning the superclasses as described next.

The overall motivation of using hierarchical classification with superclasses is to encourage more generalization by learning coarse to fine features from front to rear stacks. Also, large networks tend to overfit the training data, hence defining more tasks besides the main task can help to reduce the over-parametrization and overfitting effect to improve the performance. We expect the hierarchical classification to outperform the original multi-way classification because of the mentioned reasons, and it can be observed in Tables 6.1-6.6.
in which the hierarchical classification achieves equal or better performance in comparison with the baseline multi-way classification.

We define a graph with \( N \) nodes, each corresponding to one of the original action classes. Edge \( e_{ij} \) denotes a connection between node \( i \) and \( j \) \((i < j)\), with its cost defined as \( e_{ij} = c_{ij} + c_{ji} \), where \( c_{ij} \) is the corresponding element of the confusion matrix \( C \). We denote the assigned superclass for each action class as \( s_i^l \in \{1, \ldots, N/M_l\} \), where \( i \in \{1, \ldots, N\} \) is the action class index, \( l \in \{1, \ldots, L\} \) is the stack index, and hyperparameter \( M_l \) is the number of superclasses of stack \( l \). The bigger \( l \) is, the larger \( M_l \) is. That is, front stacks have fewer number of superclasses which only need to find the most distinguishable features to classify the actions. The rear stacks, however, need to concentrate more on finer details to differentiate actions into more categories. We also denote the set of assigned superclasses at level \( l \) as \( S_l = \{s_1^l, \ldots, s_N^l\} \). We aim to minimize the sum of edge costs among all superclasses at all levels:

\[
\forall l \in \{1, \ldots, L\} \quad \min_{S_l} \sum_{i,j \in \{1, \ldots, N\}, s_i^l \neq s_j^l} e_{ij} \quad (4.3)
\]

To obtain the superclasses we aim to solve a clustering problem. In this task action classes are points and superclasses are clusters, and all of the superclasses have equal number of action classes. In this task, the objective is to minimize the total edge costs between the superclasses. Our clustering algorithm is different from standard clustering algorithms such as k-means and k-medoids because we do not define just one center point for each superclass, and instead for each action class its edges with all the actions from other superclasses are important and computed as the total cost of the clustering and superclass computation. Therefore, multiple points or action classes are defined for each cluster or superclass in this clustering task.

We use a simple greedy algorithm to solve the clustering problem and minimize the total edge cost. We initialize the superclass assignments randomly but evenly. At each optimization step, we swap two superclass assignments that decrease the cost function the most. We continue this procedure until no more deduction can be achieved. We run the greedy optimization 1000 times and then choose the solution with the lowest cost. More advanced optimization algorithms, such as genetic algorithms or deep learning methods, can be used as well. But our early experiments showed similar performance among different algorithms. So we choose the simple greedy algorithm in the end.

The optimized superclass assignments help classify the original action classes into similar groups, which is useful in making the superclass classification in earlier levels simpler. More specifically, differentiating between similar actions in different groups is harder, and we intend to learn finer and more complex features gradually as we move towards the latter ResNet stacks. To illustrate the effect of using the confusion matrix for finding the similarity
between the action classes one example for the NTU 60 dataset is shown in Figure 4.3. All mutual classes with two people in action are classified into the same level-1 superclass. Most medical actions are in the other superclass. Figure 4.4 shows another example for the N-UCLA dataset. All class pairs in the level-3 superclass are similar; for instance, the “carry” and “walk around” classes or the “throw” and “drop trash” classes.

After the first pass of training, we denote the predicted classification probabilities by each ResNet stack as $\tilde{y}_l$. We also denote the ground truth and the superclass assignments from the greedy algorithm as $y_l$. We then train the network for a second time, using a cross-entropy loss after each stack denoted as $\text{Loss}_l(y_l, \tilde{y}_l)$. The total loss is a weighted sum of the loss for each stack:

$$\text{Loss} = \sum_l w_l \times \text{Loss}_l(y_l, \tilde{y}_l), \ l \in \{1, \ldots, L\}$$  \quad (4.4)$$

Generally speaking, we use bigger weights $w_l$ for latter stacks. This is because the prior stacks tend to extract low-level features less important for the final classification. In Figure 4.5 we show the normalized and averaged absolute gradients backpropagated from each classifier to the last convolutional layer of stack 1, when using equal weights $(1, 1, 1, 1)$ for all ResNet stacks. Note that gradients from latter stacks are richer than those of the previous stacks. We also experiment with different weighting schemes in Table 4.2 with using 2, 6, and 20 number of superclasses for the NTU RGB+D 60 dataset and 5, 20, and 80 for the Kinetics 400 dataset based on the results from Tables 5.2 and 5.3. The search space for finding the optimal loss function weights is continuous, so we chose a few schemes based on our intuition for proposing the hierarchical classification. The early experiment results for different weighting schemes were obtained by Kinetics 400 dataset, as discussed in Section 4.1.

### 4.3.1 Number of Superclass Sets

Here we discuss the size of the search space for choosing superclasses. Defining $N$ as the number of action classes, we assume that $N$ has $M$ distinct prime factors, so we can factorize the $N$ as $N = \prod_{i=1}^{M} p_i^{q_i}$, in which $p_i$s are distinct prime numbers, and $q_i$s are positive integer numbers. We denote the number of divisors of $N$ with $d$, which is calculated as:

$$d = \prod_{i=1}^{M} (q_i + 1)$$  \quad (4.5)$$

Among the divisors of $N$, we do not consider 1 nor $N$ as options for the superclass numbers because classifying for only one superclass is trivial while classifying for $N$ superclasses is equivalent to the final hierarchical classification level, which is not used for intermediate hierarchical classification levels. Therefore, we have $d - 2$ choices for the number of superclasses at each level, $L$. We have $L - 1$ hierarchical classification levels for which superclasses
should be selected because there should be \( N \) number of superclasses for the last level as it is calculating the final classification probabilities. The number of superclass sets is computed as Equation 4.6.

\[
S = (d - 2)^{L-1}
\]

(4.6)

We also assume that the number of superclasses at each level is different from the other layers, so we can compute the number of superclass sets as Equation 4.7.

\[
S = \prod_{i=0}^{L-2} (d - 2 - i)
\]

(4.7)

At last, as discussed earlier, the number of superclasses increases as we move towards the latter layers to make the classifications more complex gradually. Considering this constraint, only one combination out of each \((L - 1)!\) of superclass numbers is acceptable, so we have the Equation 4.8 to compute the number of valid superclass selections.

\[
S = \frac{\prod_{i=0}^{L-2} (d - 2 - i)}{(L - 1)!}
\]

(4.8)

The number of superclass selections computed in Equation 4.8 can be shown with binomial coefficient from Combinatorics. The Equation 4.9 shows the final calculation formula for \( S \).

\[
S = \binom{d - 2}{L - 1} = \frac{(d - 2)!}{(L - 1)! \times (d - L - 1)!} = \frac{\prod_{i=0}^{L-2} (d - 2 - i)}{(L - 1)!}
\]

(4.9)

Following these calculations, we can determine the size of the valid superclass selection search space. We have 286, 120, and 364 valid superclass selections for the Kinetics 400, NTU-RGB+D 60, and 120 datasets, respectively. Considering the large set of valid selection options, checking all combinations is computationally expensive and very time consuming, particularly when taking into account that the already large search space size gets even larger when finding optimal hierarchical loss function weights is also considered.

Table 4.2: Different weighting schemes of the hierarchical loss function on the NTU 60 and Kinetics 400 datasets.

| Superclass weights | NTU RGB+D 60 | Kinetics 400 |
|--------------------|--------------|--------------|
|                    | X-Subject    | X-View       | Top1     |
| \( \frac{1}{64}, \frac{1}{16}, \frac{1}{4}, 1 \) | 95.36% | 98.29% | 63.10% |
| \( \frac{1}{7}, \frac{1}{5}, \frac{1}{3}, 1 \) | 95.07% | 98.41% | 63.13% |
| \( \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1 \) | 95.45% | 98.59% | 63.07% |
| 1, 1, 1, 1 | 95.31% | 98.01% | 61.55% |
| 1, \( \frac{1}{7}, \frac{1}{5}, \frac{1}{3}, \frac{1}{1} \) | 93.91% | 97.20% | 56.51% |
| 1, \( \frac{1}{7}, \frac{1}{5}, \frac{1}{3}, \frac{1}{8} \) | 93.22% | 97.24% | 54.76% |
| 1, \( \frac{1}{4}, \frac{1}{16}, \frac{1}{64} \) | 92.21% | 95.85% | 52.64% |
4.4 Network Pruning

We apply network pruning to further improve the accuracy and generalization ability of our model. We also tried to apply dropout and used 50% dropout before the fully connected layers similar to [18] and lower ratio of 10% dropout after each Rectified Linear Unit (ReLU) activation layer, which comes after the convolutional layers similar to [38], but did not find it beneficial in our early experiments shown in Table 4.3. Besides, unlike network pruning, dropout can not be used for shrinking the neural network size to make it more suitable for smaller devices. We employ two iterative pruning methods while keeping the hierarchical classification approach intact. We use inherited weights instead of reinitializing the networks because we did not find retraining from scratch with reinitialized weights beneficial in our early experiments as shown in Table 4.4.

Table 4.3: Comparing our network model with network pruning and dropout on the NTU RGB+D 60 dataset. Here the full model refers to the inflated ResNet50 trained with the hierarchical loss using cropped video input.

| Method                        | X-Subject | X-View |
|-------------------------------|-----------|--------|
| full model                    | 95.45%    | 98.59% |
| full model + dropout          | 94.83%    | 97.84% |
| full model + network pruning  | 95.66%    | 98.79% |

4.4.1 Network Pruning Methods

For both pruning methods, we find the p% of the convolutional filters with the lowest $l_2$ norm and zero them out. The first method prunes the bottom p% of filters in all convolutional layers of the ResNet stacks; while the second method prunes the bottom p% of filters in each layer of the ResNet stacks independently. We compare different choices of p values on

Table 4.4: Comparing reinitializing weights with inherited weights for iterative pruning on the NTU RGB+D 60 dataset. The full model refers to the baseline network trained with the hierarchical loss using cropped video input.

| Method                        | Reinitializing Weights | Inheriting Weights |
|-------------------------------|------------------------|--------------------|
|                               | X-Subject | X-View | X-Subject | X-View |
| full model                    | 95.45%    | 98.59% | 95.45%    | 98.59% |
| full model + 1-pass pruning   | 94.60%    | 97.70% | 95.37%    | 98.66% |
| full model + 2-pass pruning   | 94.61%    | 97.99% | 95.50%    | 98.63% |
| full model + 3-pass pruning   | –         | –      | 95.64%    | 98.74% |
| full model + 4-pass pruning   | –         | –      | 95.14%    | 98.79% |
| full model + 5-pass pruning   | –         | –      | 95.39%    | 98.56% |
| full model + 5-pass pruning   | –         | –      | –         | 98.62% |


Table 4.5: Different pruning ratios and passes on the NTU 60 cross-subject benchmark.

| Pruning passes                  | 5% pruning | 10% pruning | 15% pruning |
|---------------------------------|------------|-------------|-------------|
| full model                      | 95.45%     | 95.45%      | 95.45%      |
| full model + 1-pass pruning     | 94.84%     | 95.52%      | 95.08%      |
| full model + 2-pass pruning     | 95.17%     | 95.61%      | 95.21%      |
| full model + 3-pass pruning     | –          | 95.60%      | –           |
| full model + 4-pass pruning     | –          | 95.66%      | –           |
| full model + 5-pass pruning     | –          | 95.41%      | –           |
| full model + 6-pass pruning     | –          | 95.47%      | –           |

the NTU 60 dataset using the first pruning method in Table 4.5. We will use 10% for all our experiments in Section 5. The pruning can be carried out iteratively for multiple passes. We perform pruning until we observe reductions in testing performance for two consecutive steps. We will show more pruning results for up to seven passes in Tables 6.1-6.6 in our ablation studies.

4.4.2 Network Pruning Importance

According to the Table A.1 in Appendix, more than 98% of Inflated ResNet50 learnable weights come from 3D convolutional layers. Therefore, with one or multiple levels of network pruning, we can greatly reduce the network size, which is very helpful for smaller devices such as mobile phones or smart security cameras.

4.5 Other Auxiliary Methods

In addition to the mentioned methods, we also use other auxiliary methods, which are helpful for improving the generalization and training process of our network. More specifically, we use adaptive learning rate and shuffle the training set at each epoch. For data augmentation, we flip training videos horizontally with 50% chance.

4.6 Fusing Video-based and Skeleton-based Models

Here, first, we briefly discuss the skeleton-based classification method proposed by our research group [58], which is not published yet, as we have fused this model with our video-based action classification model to boost the performance further for the two biggest datasets tested among the five. Second, we describe the fusion methods used for taking advantage of both methods.
Table 4.6: Fusion results on NTU 120 and Kinetics 400. The first two methods are the baseline models, and the next three are fusion results by weighted averaging. The last row is classification using concatenated features from both models with an MLP.

| Method                        | NTU 120 | Kinetics 400 |
|-------------------------------|---------|--------------|
|                               | X-Subject | X-Setup | Top-1 | Top-5 | Top-1 | Top-5 |
| video-based model             | 93.69%  | 94.63% | 71.63% | 88.97% |
| skeleton-based model [58]     | 83.8%   | 85.7% | 32.1% | 54.2% |
| weighted averaging ($\frac{1}{2}, \frac{1}{2}$) | 94.45% | 95.45% | 71.68% | 88.98% |
| weighted averaging ($\frac{1}{3}, \frac{1}{3}$) | 94.09% | 95.09% | 71.45% | 89.57% |
| weighted averaging ($\frac{1}{3}, \frac{1}{3}$) | 93.87% | 95.07% | 71.07% | 87.84% |
| combined features             | 92.52%  | 93.47% | 54.22% | 78.62% |

4.6.1 Skeleton-based Classification Method

The utilized skeleton-based action classification follows the GCN method and has two major components: Direction-Invariant Features (DIF) and Discriminative Feature Learning (DFL). In DIF, joint positions are transformed into the human body local coordinate system, which removes the character facing direction from the skeleton data. DIF makes motions in the same class performed in different directions become more similar and helps in achieving more generalization. In this model, DFL is added as an auxiliary branch and is supplemented with the global average pooling network proposed in [56]. DFL captures discriminative features by segmenting the skeleton data over time and selecting the most informative ones to be used for classification.

4.6.2 Fusion Methods

It is common to fuse and use both of video and skeleton human action classification methods to boost the performance [21, 40, 43, 62]. We similarly boost the performance by using a skeleton-based model, which is explained briefly in Subsection 4.6.1. We experiment with two well-known and common late fusion methods as shown in Table 4.6. The first one uses weighted averaging to combine the classification probabilities from both models. The second one combines features before the final classification layer from both models to train a Multilayer Perceptron (MLP) with two fully connected layers using ReLU activation. A third option, that we did not experiment with, is to retrain both models and the MLP at the same time. We found that the second method tends to overfit the features instead of learning to generalize over the important ones. Therefore, best results are obtained with the first method, which was also used in [21, 40, 43, 62]. Figure 4.6 illustrates our methods for fusing the two models.
Figure 4.3: NTU 60: All 60 action classes and the derived superclasses.

Figure 4.4: N-UCLA: All 10 action classes and the derived superclasses.
Figure 4.5: Averaged and normalized absolute gradients in the last convolutional layer of stack 1, backpropagated from different levels of classifiers. The gradients are computed from the first 500 clips of the NTU 60 dataset. Pure red indicates 1 and pure blue indicates 0.

Figure 4.6: Example video frames and Skeleton data. The first rows of videos and Skeleton data are from NTU RGB+D 60 dataset, and the second rows are from Kinetics 400 dataset. Fusion can be a simple weighted averaging or an MLP.
Chapter 5

Experiments

5.1 Implementation Details

Our network is based on the Inflated ResNet50 as described in Subsection 4.2.1. We train our network for 20 epochs, with an adaptive learning rate initially set to 0.0001. We divide the learning rate by 10 when the validation loss has not improved for two consecutive epochs. We use a pruning ratio of 10\% at each pruning pass.

For the experiments based on Inflated ResNet50 network, we preprocess all input videos, except Kinetics 400, with cropping and rescaling as described in Section 4.1. In addition, we flip all the video frames horizontally with a probability of 50\%. For both training and testing, we partition each video clip into eight equal segments and uniformly sample eight frames as the input frames, with the location of the first frame randomly chosen within the first segment. For training we also randomly crop a fixed $224 \times 224$ square from all input frames to feed into our network, but at test time, we crop a square of the same size at the center of each frame. We also shuffle the training data at the beginning of each epoch. For testing, multiple sets of frames are sampled from the eight segments of each clip as input to the network, and the final classification probabilities are averaged similar to [2]. For the Kinetics 400, we sample 30 sets of frames similar to [12], and we sample 5 sets of frames for all the other datasets, which is the same as [2]. 10\% of the test data is randomly chosen as the validation data.

For the second network, \textit{i.e.}, the SlowFast network [12], we do not train the hierarchical classification model from scratch but retrain our network using the saved model in the original training for computational efficiency. For both hierarchical and network pruning, we train our network for 10 more epochs, with two scheduled learning rates of 1e-6 and 1e-7. We do not change other settings of the SlowFast ResNet50 multi-grid training proposed by [12]. In this setting, the slow path receives 8 frames out of the 64 sampled frames, and the fast path has all the 64 frames as the input.
### Table 5.1: Different Inflated ResNet Architectures on NTU 60.

| Architecture  | #Parameters | X-Subject | X-View |
|---------------|-------------|-----------|--------|
| ResNet18      | 33.7M       | 93.83%    | 97.87% |
| ResNet34      | 64.0M       | 93.91%    | 97.69% |
| ResNet50      | 46.8M       | **95.25%**| **98.34%**|
| ResNet101     | 85.8M       | 95.19%    | 98.09% |

### 5.2 Network Architecture

For our first base network, we perform experiments to search for the best Inflated ResNet architecture as shown in Table 5.1. The original ResNet has five variations: ResNet18, ResNet34, ResNet50, ResNet101 and ResNet152. We do experiments on the inflated version of the first four variations, as ResNet152 is unnecessarily big for our multi-modal datasets.

We note that the Inflated ResNet34 has more parameters than the Inflated ResNet50, as they are made of different building blocks as shown in Figure 3.5. This is because ResNet34 has twice the number of 2D convolutional layers with the kernel size of 3×3, as the ResNet50. These convolutional layers are converted to 3D ones with the kernel size of 3×3×3, which makes the Inflated ResNet34 larger than the Inflated ResNet50. From Table 5.1 we conclude that the Inflated ResNet50 is the best baseline model to use for our further experiments.

### 5.3 Hyperparameters

The number of superclasses for each level of the hierarchical classification is one important hyperparameter of our model. However, when the original number of action classes is big, such as 60 for NTU 60, there are too many combinations of these parameters for all the classification levels. We thus chose some representative combinations to test for NTU 60 and the results are shown in Table 5.2. According to the results obtained on the NTU 60 dataset, we chose some combinations similar to the best one achieved for the NTU 60 to perform early experiments on the Kinetics 400 shown in Table 5.3 as discussed in Section 4.1. We have also performed experiments on different weighting schemes for the hierarchical loss function in Table 4.2, and different pruning ratios in Table 4.5.

### 5.4 Comparisons

We evaluate our method and compare with other state-of-the-art methods on five commonly used datasets: NTU RGB+D 60 Dataset (NTU 60), NTU RGB+D 120 Dataset (NTU 120), Northwestern-UCLA Multiview Action 3D Dataset (N-UCLA), UTD Multimodal Human Action Dataset (UTD-MHAD), and Kinetics 400 Dataset. For all these datasets, except Kinetics 400, we crop the videos in a preprocessing stage as described in Section 4.1. In all
Table 5.2: Different superclass sizes for level 1-3 classification on the NTU 60 dataset.

| Superclasses | X-Subject | X-View |
|--------------|-----------|--------|
| 2/6/20       | 95.45%    | 98.59% |
| 3/10/30      | 95.25%    | 98.34% |
| 4/12/30      | 95.25%    | 98.38% |
| 5/15/30      | 95.20%    | 98.29% |
| 6/12/30      | 95.39%    | 98.28% |
| 10/20/30     | 95.04%    | 98.38% |
| 60/60/60     | 95.21%    | 98.37% |

Table 5.3: Different superclass sizes for level 1-3 classification on the Kinetics 400 dataset.

| Superclasses | Top1       | Top5       |
|--------------|------------|------------|
| 4/16/80      | 62.29%     | 84.20%     |
| 4/20/80      | 62.98%     | 84.37%     |
| 5/20/80      | **63.07%** | 84.00%     |
| 5/20/100     | 62.42%     | 84.14%     |
| 5/25/100     | 62.45%     | 84.10%     |

experiments, we set the number of superclasses heuristically based on testings as described in Section 5.3. More dataset-specific settings are given in the dataset descriptions next.

### 5.4.1 Datasets

**NTU RGB+D 60 Dataset (NTU 60)** contains more than 56000 video clips [41]. 2D and 3D skeleton data, as well as depth, are also available. There are 60 action classes, and two evaluation benchmarks: cross-view and cross-subject. We choose 2, 6, 20 as the number of superclasses for levels 1-3 classification, respectively, according to Table 5.2. We use weights $(\frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1)$ for the hierarchical loss function, according to Table 4.2.

Table 5.4: Comparison on NTU 60. – indicates no results available.

| Method          | Year | Pose | RGB | X-View | X-Subject |
|-----------------|------|------|-----|--------|-----------|
| Glimpse Clouds [2] | 2018 | ✓    | ✓   | 93.2%  | 86.6%     |
| FGCN [57]       | 2020 | ✓    | ✓   | 96.25% | 90.22%    |
| MS-G3D Net [36] | 2020 | ✓    | ✓   | 96.2%  | 91.5%     |
| PoseMap [33]    | 2018 | ✓    | ✓   | 95.26% | 91.71%    |
| MMTM [21]       | 2019 | ✓    | ✓   | –      | 91.99%    |
| Action Machine [62] | 2019 | ✓    | ✓   | 97.2%  | 94.3%     |
| PGCN [43]       | 2019 | ✓    | ✓   | –      | **96.4%** |
| ours [10]       | 2020 | ✓    | ✓   | **98.79%** | 95.66%    |
Table 5.5: Comparison on NTU 120. * indicates results obtained from author-released code. – indicates no results available. Our fusion results are obtained by fusing the skeleton-based and the video-based action classification models.

| Method                          | Year | Pose | RGB | X-Subject | X-Setup |
|---------------------------------|------|------|-----|-----------|---------|
| Action Machine [62]             | 2019 | ✓    | ✓   | –         | –       |
| TSRJI [3]                       | 2019 | ✓    | –   | 67.9%     | 62.8%   |
| PoseMap from Papers with Code [32] | 2018 | ✓    | ✓   | 64.6%     | 66.9%   |
| SkeleMotion [4]                 | 2019 | ✓    | –   | 67.7%     | 66.9%   |
| GVFE + AS-GCN with DH-TCN [37]  | 2019 | ✓    | –   | 78.3%     | 79.8%   |
| Glimpse Clouds [2]              | 2018 | ✓    | –   | 83.52%*   | 83.84%* |
| FGCN [57]                       | 2020 | ✓    | –   | 85.4%     | 87.4%   |
| MS-G3D Net [36]                 | 2020 | ✓    | –   | 86.9%     | 88.4%   |
| ours [10]                       | 2020 | ✓    | ✓   | 93.69%    | 94.54%  |
| our fusion results              | 2020 | ✓    | ✓   | 94.45%    | 95.45%  |

**NTU RGB+D 120 Dataset (NTU 120)** adds 60 new action classes to the original NTU 60 dataset [30]. It contains more than 114000 video clips in total and provides two benchmarks: cross-setup and cross-subject. As the number of action classes is doubled compared with that of NTU 60, we use 4, 12, 40 as the number of superclasses for levels 1-3 classification. We still use \((\frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1)\) for weighting the hierarchical loss function.

**Northwestern-UCLA Multiview Action 3D (N-UCLA)** contains 1494 video sequences, together with depth and 3D skeleton data [50]. Each action is recorded simultaneously with three Kinect cameras. We convert the 3D skeleton data into 2D skeletons using the projection method described in Section 4.1. We use three view-based benchmarks where each view is used for testing and the other two for training. There are 10 different actions in this dataset. We thus choose 2 and 5 as the number of superclasses for the level-2 and level-3 classifiers, respectively. We do not need the level-1 classifier for hierarchical classification anymore. We use weights \((\frac{1}{16}, \frac{1}{4}, 1)\) for the hierarchical loss function. As the size of this dataset is small, we use the pre-trained network on NTU 60 cross-subject benchmark to initialize the network training on N-UCLA.

**UTD Multimodal Human Action Dataset (UTD-MHAD)** contains 861 video sequences, together with depth, 3D skeleton, and wearable inertial sensor data [9]. It provides one cross-subject evaluation benchmark. Similar as for the N-UCLA dataset, we convert the 3D skeleton data into 2D skeletons by projection and use the pre-trained network on NTU 60 for initialization. There are 27 action classes so we choose 3 and 9 as the number of superclasses for the level-2 and level-3 classifiers, respectively. We also use weights \((\frac{1}{16}, \frac{1}{4}, 1)\) for the hierarchical loss function, similar as for the N-UCLA dataset.

**Kinetics 400 Dataset (Kinetics)** contains more than 300000 labeled videos from YouTube [23]. The video clips all last 10 seconds. For cropping, we use the Kinetics-Skeleton dataset provided by [56], which is extracted skeletons by running OpenPose [5] on Kinetics.
Table 5.6: Comparison on N-UCLA. – indicates no results available. The Pre-trained column indicates if the model was pre-trained on ImageNet and/or a bigger human action dataset.

| Method                  | Year | Pre-trained | Pose | RGB | View1/View2/View3/Average |
|-------------------------|------|-------------|------|-----|--------------------------|
| PoseMap [33]            | 2018 | ✓           | ✓    | ✓   | – / – / – / –            |
| Ens. TS-LSTM[24]        | 2017 | ✓           | –    | –   | – / – / 89.22% / –       |
| EleAtt-GRU [60]         | 2018 | ✓           | ✓    | ✓   | – / – / 90.7% / –        |
| Enhanced Viz. [34]      | 2017 | ✓           | ✓    | ✓   | – / – / 92.61% / –       |
| Glimpse Clouds [2]      | 2018 | ✓           | ✓    | ✓   | 83.4% / 99.5% / 90.1% / 87.6% |
| FGCN [57]               | 2020 | ✓           | –    | –   | – / – / 95.3% / –        |
| Action Machine [62]     | 2019 | ✓           | ✓    | ✓   | 88.3% / 92.2% / 96.5% / 92.3% |
| ours [10]               | 2020 | ✓           | ✓    | ✓   | 91.10% / 91.95% / 98.92% / 93.99% |

Table 5.7: Comparison on UTD-MHAD. * indicates results obtained from author-released code. The Pre-trained column indicates if the model was pre-trained on ImageNet and/or a bigger human action dataset.

| Method                                | Year | Pre-trained | Pose | RGB | X-Subject |
|---------------------------------------|------|-------------|------|-----|-----------|
| Glimpse Clouds [2]                    | 2018 | ✓           | ✓    | ✓   | 84.19%*   |
| JTM [51]                              | 2016 | ✓           | ✓    | ✓   | 85.81%    |
| Optical Spectra [19]                  | 2018 | ✓           | ✓    | ✓   | 86.97%    |
| JDM [25]                              | 2017 | ✓           | ✓    | ✓   | 88.10%    |
| Action Machine Archived Version [61]  | 2019 | ✓           | ✓    | ✓   | 92.5%     |
| PoseMap [33]                          | 2018 | ✓           | ✓    | ✓   | 94.51%    |
| ours [10]                             | 2020 | ✓           | ✓    | ✓   | 91.63%    |
Table 5.8: Comparison on Kinetics 400 using video-based classification.* indicates results obtained from author-released code. – indicates no results available. Our fusion results are obtained by fusing the skeleton-based and the Inflated ResNet50 video-based action classification models.

| Method                                      | Year | Top-1   | Top-5   |
|---------------------------------------------|------|---------|---------|
| Action Machine [62]                         | 2019 | –       | –       |
| Glimpse Clouds [2]                          | 2018 | 44.47*  | 70.17*  |
| 2s-AGCN [44]                                | 2019 | 36.1%   | 58.7%   |
| DGNN [42]                                   | 2019 | 36.9%   | 59.6%   |
| MS-G3D Net [36]                             | 2020 | 38.0%   | 60.9%   |
| SlowFast ResNet50 + Multi-grid [12]         | 2020 | 76.60%  | 92.70%  |
| SlowFast [12]                               | 2019 | 79.8%   | 93.9%   |
| WR(2+1)D-152[15]                            | 2019 | 82.8%   | 95.3%   |
| irCSN [11]                                  | 2020 | 83.6%   | 96.0%   |
| ours with Inflated ResNet50                 | 2020 | 71.63%  | 88.97%  |
| our fusion results                          | 2020 | 71.68%  | 88.98%  |
| ours with SlowFast                          | 2020 | 76.62%  | 92.75%  |

400. There are 400 action classes, and we choose 5, 20, 80 as the number of superclasses for levels 1 to 3 classification, respectively, according to experiments similar to Table 5.2. We use \( \left( \frac{1}{27}, \frac{1}{9}, \frac{1}{3}, 1 \right) \) for the hierarchical loss function according to Table 4.2. For this dataset, we also report the top-5 accuracy as the top-1 accuracy is not very high.

5.4.2 Performance

Tables 5.4-5.8 show the comparison results. Our method scores the highest or close to the highest for all four multi-model datasets. We report the accuracy for prior work by either directly taking numbers reported in the original paper, or running author-released code if relevant performance was not reported in the original papers. For fair comparisons, we check if a method uses RGB input or pose input or both. We also mark if a method uses pre-trained models on ImageNet and/or a bigger human action dataset to initialize the network for training on small datasets.
Chapter 6

Ablation Study

We detail the performance gains of the video cropping preprocess, the hierarchical classification, and the multipass network pruning components with ablation studies shown in Tables 6.1-6.6. The hierarchical classification is proved to be beneficial in all cases after cropping. Without cropping, the hierarchical classification alone is also able to gain performance as shown in the third row of Table 6.2 for NTU 120 and in Tables 6.5 and 6.6 for Kinetics 400 dataset.

We continue the network pruning iterations until we observe lowered performance from the best one achieved so far in two consecutive steps. The network pruning is helpful in most cases. For small datasets, pruning layer by layer outperforms pruning all layers together. For larger datasets such as the NTU datasets, the two pruning methods perform comparably.

The cropping preprocess yields large performance gains in most cases, but not for the UTD-MHAD dataset, which was captured in one single environment with actors located in the frame centers already. Figure 6.1 shows a few example video frames from this dataset.
Table 6.1: Ablation Study on NTU 60 dataset. The baseline model refers to the Inflated ResNet50 network trained without hierarchical loss using the original videos. The full model refers to the baseline network trained with the hierarchical loss using cropped video input.

| Method                  | Pruning Altogether | Pruning by Layers |
|-------------------------|--------------------|-------------------|
|                         | X-Subject | X-View  | X-Subject | X-View  |
| baseline                |           |         |           |         |
| baseline + cropping     |           |         |           |         |
| full model              |           |         |           |         |
| full model + 1-pass pruning |       |         |           |         |
| full model + 2-pass pruning |       |         |           |         |
| full model + 3-pass pruning |       |         |           |         |
| full model + 4-pass pruning |       |         |           |         |
| full model + 5-pass pruning |       |         |           |         |
| full model + 6-pass pruning |       |         |           |         |

Table 6.2: Ablation Study on NTU 120 dataset.

| Method                  | Pruning Altogether | Pruning by Layers |
|-------------------------|--------------------|-------------------|
|                         | X-Subject | X-Setup | X-Subject | X-Setup |
| baseline                |           |         |           |         |
| baseline + cropping     |           |         |           |         |
| baseline + hierarchical loss |       |         |           |         |
| full model              |           |         |           |         |
| full model + 1-pass pruning |       |         |           |         |
| full model + 2-pass pruning |       |         |           |         |
| full model + 3-pass pruning |       |         |           |         |
| full model + 4-pass pruning |       |         |           |         |
| full model + 5-pass pruning |       |         |           |         |

Table 6.3: Ablation Study on N-UCLA dataset. The “Viewn” columns indicate the camera view used for testing and the other two views are used for training.

| Method                  | View1 | View2 | View3 | View1 | View2 | View3 |
|-------------------------|-------|-------|-------|-------|-------|-------|
| baseline                | 77.76%| 70.73%| 97.00%| 77.76%| 70.73%| 97.00%|
| baseline + cropping     | 88.78%| 89.00%| 98.50%| 88.78%| 89.00%| 98.50%|
| full model              | 88.78%| 89.98%| 98.92%| 88.78%| 89.98%| 98.92%|
| full model + 1-pass pruning | 89.75%| 86.64%| 98.28%| 88.39%| 88.39%| 98.71%|
| full model + 2-pass pruning | 90.14%| 89.00%| 98.28%| 91.10%| 88.39%| 98.07%|
| full model + 3-pass pruning | 71.95%| 84.36%| 84.36%| 86.85%| 91.36%| –     |
| full model + 4-pass pruning | 40.62%| –     | –     | 85.88%| 90.37%| –     |
| full model + 5-pass pruning | –     | –     | –     | 91.95%| –     | –     |
| full model + 6-pass pruning | –     | –     | –     | –     | 90.37%| –     |
| full model + 7-pass pruning | –     | –     | –     | –     | 88.21%| –     |
Table 6.4: Ablation Study on UTD-MHAD dataset.

| Method           | Pruning Altogether | Pruning by Layers |
|------------------|--------------------|-------------------|
| baseline         | 84.88 %            | 84.88 %           |
| baseline + hierarchical | 85.58 %      | 85.58 %           |
| baseline + cropping | 82.55 %       | 82.55 %           |
| full model       | **86.05%**        | 86.05%            |
| full model + 1-pass pruning | 84.65 %   | 87.44%            |
| full model + 2-pass pruning | 52.23 %   | 86.51%            |
| full model + 3-pass pruning | –          | 90.46%            |
| full model + 4-pass pruning | –          | **91.63%**        |
| full model + 5-pass pruning | –          | 90.23%            |
| full model + 6-pass pruning | –          | 90.93%            |

Table 6.5: Ablation Study on Kinetics 400 dataset with our methods and Inflated ResNet50 base network. Here, full model does not contain the video crop preprocessing stage.

| Method           | Pruning Altogether | Pruning by Layers |
|------------------|--------------------|-------------------|
| baseline         | 69.29%             | 69.29%            |
| full model       | **71.63%**         | **71.63%**        |
| full model + 1-pass pruning | 69.67 %   | **71.34%**        |
| full model + 2-pass pruning | 70.14%   | **70.86%**        |

Table 6.6: Ablation Study on Kinetics 400 dataset with our methods and SlowFast base network. Here, full model does not contain the video crop preprocessing stage.

| Method           | Pruning Altogether | Pruning by Layers |
|------------------|--------------------|-------------------|
| baseline         | 76.60 %            | 76.60 %           |
| full model       | **76.62 %**        | **76.62 %**       |
| full model + 1-pass pruning | 71.87 %   | **72.21%**        |
| full model + 2-pass pruning | 65.21%   | **67.86%**        |
Figure 6.1: Example frames from UTD-MHAD dataset. Each row contains frames from a single video and their actions are as follows. a) right arm swipe to the right b) right hand draw x c) bowling (right hand) d) two hand push e) stand to sit.
Chapter 7

Conclusion

In this work, we have augmented the Inflated ResNet50 architecture with hierarchical classification, iterative network pruning, and skeleton-based cropping and combined our model with a skeleton-based classification model to improve the performance even further. These components are simple to implement and effective in improving the human action classification accuracy. It is also valuable to mention that with our proposed methods, we have focused on improving accuracy and generalization while trying to reduce the network size. With network pruning, we make our network smaller, which can also improve the testing accuracy in some cases as well. Besides, our hierarchical classification is not needed in the testing time, so it also does not increase the network size to keep the network suitable for smaller devices. Our work has set up a new baseline for the NTU 120 dataset, which is the largest dataset of its kind.

7.1 Limitations

Although our proposed methods are simple, we do believe that they have merit since they are easy to implement and can be applied in the human action classification for improving accuracy, especially for smaller datasets. Our methods can be used to augment related researches in this field, as the backbone networks are similar to what we have used for our research.

Our choices of hyperparameters, such as the number of superclasses for each classification level, number of iterations for the pruning, and weights in the hierarchical loss function, are all set by either heuristics and/or manual searching. There is no guarantee that these parameters are optimal. Automatic search algorithms that are not computationally prohibitive are desirable.

7.2 Future Work

We have mainly focused on NTU RGB+D type of datasets that were captured in lab settings together with 2D/3D skeletons of reasonable quality. For future work, we would
like to incorporate more researches on real-world recorded datasets, which contain both indoor and outdoor videos and are obtained by collecting the uploaded videos by the users in social media, such as the Kinetics 400 dataset [23] and its two newer variations Kinetics 600 [6] and Kinetics 700 [7]. 2D skeletons extracted by pose estimation algorithms such as OpenPose [5] are also less trustworthy. Large-scale pre-training using super large neural network models is likely needed as suggested by [11, 15].

To further improve the performance, additional inputs, such as optical flow or depth frames, and combining GCN networks with our current network similar to [43] might be considered. The other option would be to use different Network Pruning techniques such as reinitializing the weights in each fine-tuning and pruning iteration.

In addition, the constraint of having equal classes in each superclass for every level might be removed, and its subsequent effects on the generalization and performance can be investigated. Moreover, the hierarchical classification can be enhanced by increasing the number of levels and using more intermediate layers to achieve more superclass classification levels. Besides, for future work in combining both of the video-based and skeleton-based action classification models, we would like to test other fusion methods, such as Support Vector Machines (SVM) or retraining both of the models with together.
Bibliography

[1] Gioia Ballin, Matteo Munaro, and Emanuele Menegatti. Human action recognition from rgb-d frames based on real-time 3d optical flow estimation. In Antonio Chella, Roberto Pirrone, Rosario Sorbello, and Kamilla Rún Jóhannsdóttir, editors, Biologically Inspired Cognitive Architectures 2012, pages 65–74. Springer Berlin Heidelberg, 2013.

[2] Fabien Baradel, Christian Wolf, Julien Mille, and Graham W. Taylor. Glimpse clouds: Human activity recognition from unstructured feature points. IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[3] Carlos Caetano, Francois Bremond, and William Robson Schwartz. Skeleton image representation for 3d action recognition based on tree structure and reference joints. SIBGRAPI Conference on Graphics, Patterns and Images, 2019.

[4] Carlos Caetano, Jessica Sena, Francois Bremond, Jefersson A. Dos Santos, and William Robson Schwartz. Skelemotion: A new representation of skeleton joint sequences based on motion information for 3d action recognition. IEEE International Conference on Advanced Video and Signal Based Surveillance, 2019.

[5] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[6] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. arXiv, arXiv:1808.01340, 2018.

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

[8] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[9] C. Chen, R. Jafari, and N. Kehtarnavaz. Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In IEEE International Conference on Image Processing, pages 168–172, 2015.

[10] Mahdi Davoodikakhki and KangKang Yin. Hierarchical action classification with network pruning. arXiv, 2007.15244, 2020.

[11] Haodong Duan, Yue Zhao, Yuanjun Xiong, Wentao Liu, and Dahua Lin. Omni-sourced webly-supervised learning for video recognition. arXiv, arXiv:2003.13042, 2020.
[12] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. *IEEE International Conference on Computer Vision*, 2019.

[13] Christoph Feichtenhofer, Axel Pinz, and Richard P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[14] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[15] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[16] Chunhui Gu, Chen Sun, David A. Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. Ava: A video dataset of spatio-temporally localized atomic visual actions. *arXiv*, arXiv:1705.08421, 2017.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[18] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv*, arXiv:1207.0580, 2012.

[19] Y. Hou, Z. Li, P. Wang, and W. Li. Skeleton optical spectra-based action recognition using convolutional neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(3):807–811, 2018.

[20] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018.

[21] Hamid Reza Vaezi Joze, Amirreza Shaban, Michael L. Iuzzolino, and Kazuhito Koishida. Mntm: Multimodal transfer module for cnn fusion. *arXiv*, arXiv:1911.08670, 2019.

[22] Jeremy Kawahara and Ghassan Hamarneh. Multi-resolution-tract cnn with hybrid pre-trained and skin-lesion trained layers. In Li Wang, Ehsan Adeli, Qian Wang, Yinghuan Shi, and Heung-Il Suk, editors, *Machine Learning in Medical Imaging*, pages 164–171. Springer International Publishing, 2016.

[23] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv*, arXiv:1705.06950, 2017.
[24] I. Lee, D. Kim, S. Kang, and S. Lee. Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks. In *IEEE International Conference on Computer Vision*, pages 1012–1020, 2017.

[25] C. Li, Y. Hou, P. Wang, and W. Li. Joint distance maps based action recognition with convolutional neural networks. *IEEE Signal Processing Letters*, 24(5):624–628, 2017.

[26] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017.

[27] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Actional-structural graph convolutional networks for skeleton-based action recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[28] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Symbiotic graph neural networks for 3d skeleton-based human action recognition and motion prediction. *arXiv*, arXiv:1910.02212, 2019.

[29] J. Liu, G. Wang, P. Hu, L. Duan, and A. C. Kot. Global context-aware attention lstm networks for 3d action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3671–3680, 2017.

[30] Jun Liu, Amir Shahroudy, Mauricio Lisboa Perez, Gang Wang, Ling-Yu Duan, and Alex Kot Chichung. Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[31] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. *Lecture Notes in Computer Science*, page 816–833, 2016.

[32] M. Liu and J. Yuan. Recognizing human actions as the evolution of pose estimation maps. [https://paperswithcode.com/paper/recognizing-human-actions-as-the-evolution-of](https://paperswithcode.com/paper/recognizing-human-actions-as-the-evolution-of). Accessed: 2020-05-12.

[33] M. Liu and J. Yuan. Recognizing human actions as the evolution of pose estimation maps. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1159–1168, 2018.

[34] Mengyuan Liu, Hong Liu, and Chen Chen. Enhanced skeleton visualization for view invariant human action recognition. *Pattern Recognition*, 2017.

[35] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019.

[36] Ziyu Liu, Hongwen Zhang, Zhenghao Chen, Zhiyong Wang, and Wanli Ouyang. Disentangling and unifying graph convolutions for skeleton-based action recognition. *arXiv*, arXiv:2003.14111, 2020.
[37] Konstantinos Papadopoulos, Enjie Ghorbel, Djamila Aouada, and Björn Ottersten. Vertex feature encoding and hierarchical temporal modeling in a spatial-temporal graph convolutional network for action recognition. arXiv, arXiv:1912.09745, 2019.

[38] Sungheon Park and Nojun Kwak. Analysis on the dropout effect in convolutional neural networks. In Asian Conference on Computer Vision, 2016.

[39] Wei Peng, Xiaopeng Hong, Haoyu Chen, and Guoying Zhao. Learning graph convolutional network for skeleton-based human action recognition by neural searching. arXiv, arXiv:1911.04131, 2019.

[40] Juan-Manuel Perez-Rua, Valentin Vielzeuf, Stephane Pateux, Moez Baccouche, and Frederic Jurie. Mfas: Multimodal fusion architecture search. IEEE Conference on Computer Vision and Pattern Recognition, 2019.

[41] A. Shahroudy, J. Liu, T. Ng, and G. Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In IEEE Conference on Computer Vision and Pattern Recognition, pages 1010–1019, 2016.

[42] L. Shi, Y. Zhang, J. Cheng, and H. Lu. Skeleton-based action recognition with directed graph neural networks. In IEEE Conference on Computer Vision and Pattern Recognition, pages 7904–7913, 2019.

[43] Lei Shi, Yifan Zhang, Jian Cheng, and Han-Qing Lu. Action recognition via pose-based graph convolutional networks with intermediate dense supervision. ArXiv, arXiv:1911.12509, 2019.

[44] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In IEEE Conference on Computer Vision and Pattern Recognition, 2019.

[45] Chenyang Si, Wentao Chen, Wei Wang, Liang Wang, and Tieniu Tan. An attention enhanced graph convolutional lstm network for skeleton-based action recognition. IEEE Conference on Computer Vision and Pattern Recognition, 2019.

[46] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Proceedings of the International Conference on Neural Information Processing Systems - Volume 1, page 568–576. MIT Press, 2014.

[47] Yi-Fan Song, Zhang Zhang, and Liang Wang. Richly activated graph convolutional network for action recognition with incomplete skeletons. 2019 IEEE International Conference on Image Processing, 2019.

[48] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[49] Hongsong Wang and Liang Wang. Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks. IEEE Conference on Computer Vision and Pattern Recognition, pages 3633–3642, 2017.
[50] J. Wang, X. Nie, Y. Xia, Y. Wu, and S. Zhu. Cross-view action modeling, learning, and recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2649–2656, 2014.

[51] Pichao Wang, Zhaoyang Li, Yonghong Hou, and Wanqing Li. Action recognition based on joint trajectory maps using convolutional neural networks. *ACM International Conference on Multimedia*, 2016.

[52] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[53] Chao-Yuan Wu, Ross Girshick, Kaiming He, Christoph Feichtenhofer, and Philipp Krahenbuhl. A multigrid method for efficiently training video models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[54] Cinna Wu, Mark Tygert, and Yann LeCun. A hierarchical loss and its problems when classifying non-hierarchically. *PLOS ONE*, 14(12), 2019.

[55] Cong Wu, Xiao-Jun Wu, and Josef Kittler. Spatial residual layer and dense connection block enhanced spatial temporal graph convolutional network for skeleton-based action recognition. In *The IEEE International Conference on Computer Vision Workshops*, 2019.

[56] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Association for the Advancement of Artificial Intelligence*, 2018.

[57] Hao Yang, Dan Yan, Li Zhang, Dong Li, YunDa Sun, ShaoDi You, and Stephen J. Maybank. Feedback graph convolutional network for skeleton-based action recognition. *arXiv*, arXiv:2003.07564, 2020.

[58] Zeshi Yang and Kangkang Yin. Improving skeleton-based action recognition with robust spatial and temporal features. *arXiv*, 2008.00324, 2020.

[59] Yong Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1110–1118, 2015.

[60] Pengfei Zhang, Jianru Xue, Cuiling Lan, Wenjun Zeng, Zhanning Gao, and Nanning Zheng. Adding attentiveness to the neurons in recurrent neural networks. In *Proceedings of the European Conference on Computer Vision*, pages 135–151, 2018.

[61] Jiagang Zhu, Wei Zou, Liang Xu, Yiming Hu, Zheng Zhu, Manyu Chang, Junjie Huang, Guan Huang, and Dalong Du. Action machine: Rethinking action recognition in trimmed videos. *arXiv*, arXiv:1812.05770, 2018.

[62] Jiagang Zhu, Wei Zou, Zhu Zheng, Liang Xu, and Guan Huang. Action machine: Toward person-centric action recognition in videos. *IEEE Signal Processing Letters*, PP, 2019.
Appendix A

Convolutional Layers Weights

Here we compute the total convolutional layers weights in our Inflated ResNet50 network. These computations are shown in TableA.1.
Table A.1: Inflated ResNet50 convolutional and total network weights.

| Stack layer | 3D BottleNeck Block | Dimension | # Learnable Weights |
|-------------|---------------------|-----------|---------------------|
| Stack 1     | block1              | 64 x 64 x 1 x 1 | 4,096               |
|             |                     | 64 x 64 x 3 x 3 | 110,592             |
|             |                     | 64 x 256 x 1 x 1 | 16,384              |
|             |                     | 64 x 256 x 1 x 1 | 16,384              |
|             | block2/3            | 256 x 64 x 1 x 1 | 16,384              |
|             |                     | 64 x 64 x 3 x 3 | 110,592             |
|             |                     | 64 x 256 x 1 x 1 | 16,384              |
| Stack 2     | block1              | 256 x 128 x 1 x 1 | 32,768              |
|             |                     | 128 x 128 x 3 x 3 | 442,368             |
|             |                     | 128 x 512 x 1 x 1 | 65,536              |
|             |                     | 256 x 512 x 1 x 1 | 131,072             |
|             | block2/3/4          | 512 x 128 x 1 x 1 | 65,536              |
|             |                     | 128 x 128 x 3 x 3 | 442,368             |
|             |                     | 128 x 512 x 1 x 1 | 65,536              |
| Stack 3     | block1              | 512 x 256 x 1 x 1 | 131,072             |
|             |                     | 256 x 256 x 3 x 3 | 1,769,472           |
|             |                     | 256 x 1024 x 1 x 1 | 262,144             |
|             |                     | 512 x 1024 x 1 x 1 | 524,288             |
|             | block2/3/4/5/6      | 1024 x 256 x 1 x 1 | 262,144             |
|             |                     | 256 x 256 x 3 x 3 | 1,769,472           |
|             |                     | 256 x 1024 x 1 x 1 | 262,144             |
| Stack 4     | block1              | 1024 x 512 x 1 x 1 | 524,288             |
|             |                     | 512 x 512 x 3 x 3 | 7,077,888            |
|             |                     | 512 x 2048 x 1 x 1 | 1,048,576            |
|             |                     | 1024 x 2048 x 1 x 1 | 2,097,152            |
|             | block2/3            | 2048 x 512 x 1 x 1 | 1,048,576            |
|             |                     | 512 x 512 x 3 x 3 | 7,077,888            |
|             |                     | 512 x 2048 x 1 x 1 | 1,048,576            |
| Total Stacks| –                   | –           | 46,080,000           |
| Total Weights| –                  | –           | 46,795,975           |