Human pose estimation and action recognition based on monocular video inputs

Leong, Mei Chee

2019

Leong, M. C. (2019). Human pose estimation and action recognition based on monocular video inputs. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/136596>

<https://doi.org/10.32657/10356/136596>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 20:02:53 SGT

**HUMAN POSE ESTIMATION AND ACTION
RECOGNITION BASED ON MONOCULAR VIDEO INPUTS**

LEONG MEI CHEE

**Interdisciplinary Graduate School
NTU Institute for Media Innovation (IMI NTU)**

A thesis submitted to the Nanyang Technological University in
partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2019

Statement of Originality

I hereby declare that the work reported in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

27 July 2019

Mei Chee

.....
Date

.....
Leong Mei Chee

Supervisor Declaration Statement

We have reviewed the contents and presentation style of this thesis and declare it is free of plagiarism, and of sufficient clarity and writing quality to be examined. To the best of our knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. We confirm that the investigations were conducted in accordance with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

13 Aug 2019
.....
Date Assoc. Prof Lee Yong Tsui

13 Aug 2019
.....
Date Assoc. Prof Lin Feng

Authorship Attribution Statement

This thesis contains materials from one paper accepted for publication in the following peer-reviewed journal where I was the first and corresponding author.

Chapter 3 is accepted for publication as M. C. Leong, F. Lin, and Y. T. Lee, “3D human motion recovery from a single video using dense spatio-temporal features with exemplar-based approach”, *International Journal of Machine Learning and Computing (IJMLC, ISSN: 2010-3700)*.

The contributions of the co-authors are as follows:

- I planned the project, carried out the implementation and evaluation on datasets and prepared the manuscript drafts.
- Assoc. Prof Lee Yong Tsui provided project direction, advice and reviewed the manuscript draft.
- Assoc. Prof Lin Feng provided guidance and reviewed the manuscript draft.

Chapter 4 is under reviewed by IEEE Access as M. C. Leong, D. K. Prasad, Y. T. Lee and F. Lin, “A Single-stream Spatio-temporal Aggregation Network for Video-level Action Recognition”.

The contributions of the co-authors are as follows:

- I planned the project, carried out the implementation and evaluation on datasets and prepared the manuscript drafts.
- Assoc Prof Dilip Prasad provided guidance and reviewed the manuscript draft.
- Assoc. Prof Lee Yong Tsui reviewed the manuscript draft and provided comments.

- Assoc. Prof Lin Feng provided advice and reviewed the manuscript draft.

Chapter 6 is under reviewed by The 35th ACM Symposium on Applied Computing (SAC 2020) as M. C. Leong, D. K. Prasad, Y. T. Lee and F. Lin, “Semi-CNN Architecture for Effective Spatio-Temporal Learning in Action Recognition”.

The contributions of the co-authors are as follows:

- I planned the project, carried out the implementation and evaluation on datasets and prepared the manuscript drafts.
- Assoc Prof Dilip Prasad provided computational resources, guidance, technical advice and reviewed the manuscript draft.
- Assoc. Prof Lee Yong Tsui reviewed the manuscript draft and provided comments.
- Assoc. Prof Lin Feng reviewed the manuscript draft.

21 Nov 2019

Mei Chee

.....

Date

.....

Leong Mei Chee

ACKNOWLEDGEMENT

This PhD programme and research work is funded and supported by Nanyang Technological University (NTU), Singapore. I am grateful for the financial support from NTU Research Scholarship throughout my candidature from 2015 to 2019.

I would like to specially express my deepest and sincere gratitude to my main supervisor, Assoc. Prof Lee Yong Tsui, who has offered me the opportunity to work on this research, in which I have gained invaluable experience and knowledge. His continuous support, advice, patience and encouragement propelled me towards the objectives. He provided me all the freedom to work on my research and is always there whenever I needed help. I would also like to express my great appreciation to my co-supervisor, Assoc Prof Lin Feng, for his academic support, guidance and constructive suggestions that guided me all the way in my work. His valuable insight and expertise have provided clear direction for my research. I am also grateful to Assoc. Prof Cai Yiyu, as my Thesis Advisory Committee, for his constructive comments and suggestions on the work.

I am deeply indebted to Assoc. Prof Dilip Prasad from The Arctic University of Norway, for his mentorship and his generous support by allowing me to use his lab's computational resources to complete my work. His immense knowledge has helped me a lot and his encouragement has pushed me to keep going. I also greatly appreciate his job referrals and strong recommendations. I would also like to extend my gratitude to Assoc. Prof Quek Hiok Chai, for offering me a part-time job opportunity to work on a related research project, and his valuable life advice.

I would like to acknowledge the faculties and staff from my school, Interdisciplinary Graduate School (IGS), and my research center, Institute for Media Innovation (IMI), for their academic and admin support that ensures smooth completion of this thesis. My special thanks to the lab technicians in

Computer Aided Engineering Lab in providing me the equipment and resources for completing my work. To my peers, colleagues, lab mates and fellow Residential Mentors, I am greatly thankful for their precious support, continuous encouragement and helpful advice throughout this time.

Finally, I would like to express my gratitude to my family and friends, who are always there for me. Their unfailing support, understanding and care have sustained me so far and make it possible for me to strive towards this accomplishment.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	v
TABLE OF CONTENTS	vii
ABSTRACT	xi
LIST OF FIGURES	xii
LIST OF TABLES	xv
LIST OF ABBREVIATIONS	xvii
1. CHAPTER 1 INTRODUCTION	1
1.1 Background	1
1.2 Motivation	3
1.3 Scope and Objectives	4
1.4 Organization	5
2. CHAPTER 2 LITERATURE REVIEW	7
2.1 Survey on Human Motion Recovery	7
2.1.1 Taxonomies	8
2.1.1.1 Model-based Approach	9
2.1.1.2 Example-based Approach	12
2.1.1.3 Learning-based Approach	13
2.1.1.3.1 Deep learning with Convolution Neural Network	14
2.1.1.3.2 Deep Learning with Recurrent Neural Network	16
2.1.1.4 Dynamic model	17
2.1.1.5 Others	18
2.1.2 Summary	18
2.2 Survey on Human Action Recognition	19
2.2.1 Taxonomies	19
2.2.1.1 3D CNN / Spatio-temporal CNN	20
2.2.1.2 Two-stream / Multiple-stream CNN	21
2.2.1.3 Pose-based CNN	23
2.2.1.4 Recurrent Neural Network / LSTM / Hybrid approach	24
2.2.1.5 Unsupervised Approach	26
2.2.2 Summary	26

3. CHAPTER 3 3D HUMAN MOTION RECOVERY FROM A SINGLE VIDEO USING DENSE SPATIO-TEMPORAL FEATURES WITH EXEMPLAR-BASED APPROACH.....	28
3.1 Introduction.....	28
3.2 Related Work.....	29
3.3 Methodology	30
3.3.1 Subject Alignment.....	31
3.3.2 Forming Video Patches and Spatio-Temporal Blocks.....	33
3.3.3 Feature Descriptors	34
3.3.4 Pose Estimation	36
3.4 Experimental Setup and Results.....	37
3.4.1 Implementation	37
3.4.2 Evaluation on Silhouette Images	38
3.4.2.1 Experiment on different feature representation and temporal range	38
3.4.3 Evaluation on Color Images.....	42
3.4.3.1 Experiment on Subject's Alignment and Weighted Descriptors.....	42
3.5 Comparison with State-of-the-art.....	45
3.6 Conclusion	48
4. CHAPTER 4 A SINGLE-STREAM SPATIO-TEMPORAL NETWORK FOR VIDEO-LEVEL ACTION RECOGNITION	50
4.1 Introduction	50
4.2 Related Work.....	50
4.2.1 Dynamic Temporal Encoding.....	51
4.3 Proposed Framework	52
4.3.1 Model Architecture.....	52
4.3.2 Features Extraction.....	54
4.3.2.1 Spatial Features from Pretrained Model	55
4.3.2.2 3D Spatio-temporal Features	55
4.3.3 Weighted Features Aggregation	56
4.3.4 Architecture Comparison	58
4.4 Experiments.....	59
4.4.1 Implementation Details.....	59
4.4.2 Architecture Search.....	59
4.4.3 Weighted Features Representation.....	63

4.4.4	Temporal Dynamics.....	67
4.5	Comparison with State-Of-The-Art	67
4.5.1	Qualitative Results and Visualization	69
4.6	Conclusion	73
5.	CHAPTER 5 EXTENDED SPATIO-TEMPORAL NETWORK WITH TEMPORAL PYRAMID ENCODING AND GLOBAL FEATURES AGGREGATION	74
5.1	Introduction	74
5.2	Extended Framework.....	74
5.2.1	Additional 3D convolution layers.....	75
5.2.2	Global Feature Representation.....	76
5.2.3	Temporal Pyramid Encoding	80
5.3	Experiments.....	80
5.3.1	Evaluation on Two 3D Convolution Layers.....	80
5.3.2	Evaluation on Global Representations.....	88
5.3.3	Evaluation on Temporal Pyramid Encoding.....	96
5.4	Comparison with Baseline Model and Analysis	97
5.4.1	Adding 3D convolution layers and number of input frames.....	98
5.4.2	Global features aggregation.....	99
5.4.3	Temporal pyramid encoding.....	100
5.5	Features Visualization and Qualitative Results.....	100
5.6	Conclusion	105
6.	CHAPTER 6 SEMI-CNN ARCHITECTURE FOR EFFECTIVE SPATIO-TEMPORAL LEARNING IN ACTION RECOGNITION	106
6.1	Introduction	106
6.2	Related Work.....	106
6.2.1	Full 3D CNN	107
6.2.2	Spatio-temporal fusion	107
6.3	Proposed Architecture.....	108
6.3.1	Base Model.....	109
6.3.1.1	Very Deep Convolutional Networks (VGG).....	109
6.3.1.2	Residual Networks (ResNets)	110
6.3.1.3	Dense Convolutional Networks (DenseNets)	111
6.3.2	Configuration Comparison with 2D and 3D CNN.....	111

6.4	Experiments	116
6.4.1	Implementation details	116
6.4.2	Trainable Parameters an Configurations	116
6.4.3	Evaluation on validation dataset	118
6.5	Features Visualization and Qualitative Results Comparison.....	122
6.6	Conclusion	127
7.	CHAPTER 7 CONCLUSION.....	128
7.1	Research contributions	128
7.2	Directions for future work.....	131
7.3	Conclusion	133
8.	REFERENCE	135

ABSTRACT

This thesis presents the research work for the PhD program that focuses on investigating the main processes in human motion analysis from a single video camera, which includes: 1) 3D human pose estimation, 2) motion reconstruction, and 3) action recognition. Motion capture and analysis is an active research area in computer vision with a wide range of applications. However, there exist major challenges in recovering monocular human motion due to the lack of depth information. Single frame pose estimation and tracking methods have limitations in recovering failed tracking pose and occurrence of motion jitters. To address current limitations in per-frame pose estimation and tracking methods, we propose to directly estimate a sequence of poses from a stack of consecutive frames. We exploit example-based method with dense spatio-temporal features to find best matching poses and then perform interpolation to achieve smooth motion reconstruction. For action recognition task, we exploit learning-base method, specifically deep learning with Convolutional Neural Network (CNN), for effective learning of spatio-temporal features to identify different action class in high volume of video dataset. In the initial study, a number of experiments were conducted to evaluate the effectiveness of configurations in our architecture, followed by an extended study on deeper models. Lastly, we developed a generalized architecture with fusion of 1D, 2D and 3D convolution layers, that can be adopted to existing CNN models while retaining the network's learning properties. Our empirical studies demonstrated the advantages of our architecture over its corresponding 3D CNN models in: 1) a boost of 16 – 30% improved prediction accuracy, 2) effective spatio-temporal learning, and 3) lower computational cost. The future goal of this project is to link all the main processes in this research work to develop a full pipeline human motion analysis system that can be applied in real-life applications, such as healthcare or sports analysis.

LIST OF FIGURES

	Page
Figure 2.1. (a) A pictorial structure of human body [23]. (b) A deformable part model represented by a coarse global template (left), parts template (middle) and spatial model (right) [21].	10
Figure 2.2. (a) Oriented part model in traditional approach. (b) Non-oriented image patches used to approximate limbs' transformation. (c) Various spring models are used to connect image patches for different spatial arrangements [34].	11
Figure 2.3. A graph structured hierarchy pose representation. Colored rectangles denote body parts (parameterized with poselet templates) and black edges denote relation between parts [37].	12
Figure 2.4. Three types of posebit that consider: joints distance, bending angle, and between parts position [41].	12
Figure 2.5. 3D pose estimation from noisy observations. (a) Raw input image (b) Detected parts location (c) Projection of estimated 3D poses on the scene (d) Estimated 3D poses [49].	14
Figure 2.6. The architecture of convolutional network to train features for human pose estimation [51].	15
Figure 2.7. A multi-task deep learning architecture that jointly trains detection and regression of joint positions [53].	15
Figure 2.8. A stacked hourglass architecture for repeated bottom-up and top-down processing [58].	16
Figure 2.9. The architecture of a 3D CNN [77].	21
Figure 2.10. A two-stream CNN architecture to learn spatial and temporal feature for action recognition [82].	22
Figure 2.11. An architecture with pose estimation framework, and body parts spatio-temporal feature extraction. Concatenation of all body parts' feature form the descriptor P-CNN [94].	24
Figure 2.12. The framework for sequential learning structured with 3D CNN and LSTM for action classification [98].	25
Figure 3.1. Overview of the proposed 3D motion recovery in video volume using densely extracted spatio-temporal features.	31
Figure 3.2. Extraction of human silhouette. (a) Original RGB image. (b) Background image with mean chromaticity. (c) Original image represented by its chromaticity. (d) Foreground mask obtained from RGB background model with cast shadows. (e) Foreground mask obtained from chromaticity background model (f) Final silhouette after background subtraction, shadow removal and noise removal.	32
Figure 3.3. Average image gradients across all frames in a video volume. (a) Original silhouette image, (b) Aligned and centered subject (size 240×480) (c) Subject is scaled to the same height before alignment (size 400×400).	33

Figure 3.4. A video volume that contains a sequence of image frames can be split into sub-volumes. (b) A video patch from time t to $t + n$ frames. (c) Each frame is divided into areas for feature encoding.	34
Figure 3.5. A 3D video volume is divided into spatio-temporal blocks for feature extraction and encoding. (a) A single block with size $6 \times 6 \times 6$. (b) A descriptor represented by $3 \times 3 \times 2$ blocks. Descriptors contain overlapping blocks, where a single block is reused in multiple descriptors except for the corner blocks.	34
Figure 3.6. Illustration of the process flow to compute HOG, optical flow and MBH features followed by forming two descriptor representations.....	35
Figure 3.7. An illustration of matching frames in two sequences of varying speed using Dynamic Time Warping.....	36
Figure 3.8. An illustration of estimating the 3D pose for a single frame. From the test video patches, 3D poses from the best matching exemplar are stored as candidate poses for each frame. The final estimated pose is obtained by computing the mean pose from all candidate poses.....	37
Figure 3.9. Comparison of results for video patch interval of one frame and 6 frames in the training and testing dataset. Table 3.1's results are plotted in the top-most chart.....	40
Figure 3.10. Examples of reconstructed 3D poses for selected silhouette images from the dataset provided by Agarwal and Triggs [117].	41
Figure 3.11. Tracking results for torso (top) and left hip angle (bottom) of the human in the dataset provided by [117].	41
Figure 3.12. Sample frames for evaluation. a) Original image (640×480). (b) Aligned and centered subject (240×480). (c) Scaled and aligned subject with background removed (400×400).....	43
Figure 3.13. Comparison of results on HumanEva-I dataset for Subject 2's walking action.....	43
Figure 3.14. Tracking results for Subject S2 in walking action in the scaled and aligned dataset, using a weighted descriptor of $0.2 \cdot \text{HOG} + 0.5 \cdot \text{HOF} + 0.3 \cdot \text{MBH}$. (Top) Torso rotation angle (Bottom) Left hip rotation angle.....	44
Figure 3.15. Examples of best matching exemplars for video patches of three different input images. (a) Original image, (b) aligned and centered subject, (c) scaled and center aligned subject.	45
Figure 3.16. Examples of reconstructed 3D pose for subject S1, S2 and S3 for walking, jogging and boxing actions. Images on the left shows input frame with projected 2D pose estimation, while images on the right shows 3D pose estimation viewing from a different view point.....	48
Figure 4.1. The process of our single-stream architecture that captures spatial and temporal features with weighted aggregation for video-level action recognition.....	54
Figure 4.2. Local features from each video segment are used to compute a global representation, followed by element-wise multiplication to obtain an aggregated feature representation.....	57

Figure 4.3. Comparison of three single-stream architectures with RGB input: (a) spatial network [125], (b) spatio-temporal network (ours) and (c) weighted spatio-temporal network (ours).....	58
Figure 4.4. Loss and accuracy performance comparison for training (blue line) and validation (orange line) with three different kernel sizes in 3D convolution layer.....	62
Figure 4.5. Comparison of accuracy plot for local and weighted features using base model ResNet, DenseNet and BN-Inception.....	66
Figure 4.6. Examples of qualitative results and features visualization for model BN-Inception using local and weighted features.	73
Figure 5.1. Structure of 3D convolution layers for: (a) a single layer with 3 input frames, (b) two layers with 3 input frames, and (c) two layers with 5 input frames..	75
Figure 5.2. Illustrations of four global representations used: 1) mean features, 2) weighted mean features, 3) weighted features concatenation, and 4) weighted 3D convolution features.	79
Figure 5.3. Loss and accuracy plots for different configurations network with input size of 3 frames.	86
Figure 5.4. Loss and accuracy plots for different configurations network with input size of 5 frames.	88
Figure 5.5. Loss and accuracy plots for global aggregation functions applied on <i>l3 dense</i> network.....	92
Figure 5.6. Loss and accuracy plots for global aggregation functions applied on <i>l5 dense</i> network.....	96
Figure 5.7. Examples of qualitative results and features visualization for BN-Inception model with <i>l5 dense</i> configuration.	104
Figure 6.1. Variants of architecture for ResNet model as described in [137]. (a) 2D ResNnet, (b) ResNet with mixed convolutions, (c) reversed mixed convolutions, (d) 3D ResNet, (e) ResNet with (2+1)D convolutions.....	108
Figure 6.2. Architecture comparison for (a) 2D CNN, (b) 3D CNN and (c) our proposed semi-CNN.....	109
Figure 6.3. Residual learning block in ResNet [130].	110
Figure 6.4. A dense block with 5 connected layers. Features from lower layers are used as inputs for all sub-sequent layers [133].....	111
Figure 6.5. Performance plots for 3D-CNN and semi-CNN architectures applied on different models – VGG, ResNet and DenseNet.....	121
Figure 6.6. Examples of qualitative results for 3D- and Semi-ResNet-18 models.....	127

LIST OF TABLES

	Page
Table 2.1 An overview of human pose estimation taxonomies.	9
Table 2.2 Taxonomy for action recognition approaches.	20
Table 3.1 Mean error of joint angles (degrees) for the dataset from [117] with descriptor sizes $3 \times 3 \times 1$ and $3 \times 3 \times 2$. The weighted combinations for the four descriptors are given in the first four columns.	38
Table 3.2 Quantitative result comparisons by computing the mean 3D absolute errors between estimated and ground truth joint positions (in mm). Our method shows the average error obtained from the individual and combined descriptors of HOG, HOF and MBH.	47
Table 4.1 Configurations of architecture search with different kernel sizes and dimensions for 3D convolution and pooling layer.	60
Table 4.2 Validation results with top-1 accuracy and number of learnable weights for different configurations. $k1$, $k2$ and $k3$ denotes three kernel sizes - $(3 \times 1 \times 1)$, $(3 \times 3 \times 3)$ and $(3 \times 5 \times 5)$. Bold value shows the best performance.	60
Table 4.3 Comparison of validation results for different pretrained models that utilize local and weighted features.	64
Table 4.4 Classification prediction accuracy (%) for base model ResNet-34 with varying temporal dynamics. Bold values show the best performance for the different number of segments.	67
Table 4.5 Performance comparison with state-of-the-art methods on UCF-101 dataset.	69
Table 5.1 Configurations of different 3D convolution kernel sizes and pooling layer for input of 3 frames. The first layer is fixed for all configurations, and the second layer performs spatial convolution.	76
Table 5.2 Configurations of different 3D convolution kernel sizes and pooling layer for input of 5 frames. The first layer is fixed for all configurations, and the second layer performs spatio-temporal convolution.	76
Table 5.3 Number of trainable parameters for different configurations with input sizes of 3 frames and 5 frames, using base model BN-Inception.	81
Table 5.4 Validation results that shows top-1 and top-5 accuracies for configurations shown in Table 5.1 and Table 5.2.	81
Table 5.6 Number of trainable parameters for network when different global representations were used for feature aggregation.	89
Table 5.7 Validation results for different global feature aggregation functions.	89
Table 5.8 Number of trainable parameters for temporal pyramid encoding using base model BN-Inception and Resnet-34 for a single 3D convolution layer network.	97
Table 5.9 Validation results for temporal pyramid encoding using base model BN- Inception.	97

Table 5.10 Validation results for temporal pyramid encoding using base model ResNet-34.....	97
Table 5.11 Performance comparison for network using one or more 3D convolution layers and input of 3 or more frames.	98
Table 5.12 Performance comparison for network using local features only or with global features aggregation.....	99
Table 5.13 Performance comparison for network with varying temporal strides applying on base model ResNet-34.....	100
Table 6.1 Configuration comparison for VGG-16, C3D and our model semi-VGG-16.	113
Table 6.2 Configuration comparison for 2D ResNet, 3D ResNet and our semi-ResNet, for an 18-layers network.	114
Table 6.3 Configuration comparison for 2D DenseNet, 3D DenseNet and our semi-DenseNet, with network depth of 121 layers.....	115
Table 6.4 Number of trainable parameters and configuration setting for VGG-16 models.....	118
Table 6.5 Number of trainable parameters and configuration settings for ResNet models with different number of layers.....	118
Table 6.6 Number of trainable parameters and configuration settings for DenseNet models of different depth.	118
Table 6.7 Validation results on models VGG, ResNet and DenseNet for 3D-CNN and semi-CNN architectures comparison.	119

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long short-term memory
BN	Batch normalization

CHAPTER 1

INTRODUCTION

1.1 Background

Vision-based human motion capture has been an active research topic for decades owing to a wide range of applications in the areas of surveillance, control and analysis [1]. It is useful especially for sports science and clinical applications, where the reconstructed 3D human motion can be used to analyze the performance of an athlete or a patient's movements during rehabilitation [2]. The entertainment industry utilizes reconstructed human motions for games and animation in a virtual or augmented environment, while motion analysis in surveillance is useful for human identification, tracking, as well as establishment of anomalous activities.

Human motion analysis covers a number of processes, starting from human detection and extraction, 2D and 3D pose estimation, motion tracking, motion reconstruction, action recognition, followed by activity understanding. From a single image, we can perform human detection and pose estimation. When given a sequence of video frames, we can perform motion tracking using the estimated pose to reconstruct a smooth articulated motion. On the other hand, action recognition is to categorize the types of action in a full video sequence. It may or may not involve any of the previous processes to facilitate the recognition. Activity understanding allows us to study human behaviors and interactions, or to predict future actions. It can be useful for detecting anomalous activities, person identification and analyzing an athlete's performance.

Human pose estimation and motion reconstruction in real-world scenarios are extremely challenging due to the high degrees of freedom in a human's articulated kinematic structure and its variations in shape and appearance, occlusion, 3D projection ambiguity, arbitrary camera viewpoints,

and the complexity and variability of the background [3]. Human motion is highly variable with varied poses across people of different sizes and appearance. Estimation of joint angles requires extensive computation due to the high dimensional search space. Occlusion occurs when body parts are partially hidden due to camera viewpoint, interactions with people, or foreground objects. Projection ambiguities arise due to the loss of depth information in a single image, where the observed 2D pose can be mapped to multiple feasible 3D poses. Lighting changes in the environment may also affect the detection and prediction results.

With high volumes of videos for all kinds of motions, it is challenging to identify specific action classes as there are not only appearance variations but also variations in temporal dynamics. To solve the problem of action recognition, deep learning methods such as Convolutional Neural Network (CNN) [4] have demonstrated great potential in discovering underlying features given a large number of training examples. The key question in the training process for action classification is: how to extract a good motion representation which is discriminative, meaningful and robust. A representation of an action sequence can be useful for detecting and recognizing actions from video frames, and it can be extended to motion reconstruction and semantic understanding.

Even though there is an enormous amount of research in vision-based human motion analysis [1-3, 5-10], it remains an open problem to achieve reliable performance for motion reconstruction and recognition. Extracting robust and discriminative representations with a practical computational cost is not trivial due to the challenges mentioned above. This heightens our attention and leads us to investigate the processes in human motion analysis, specifically on: 1) 3D pose estimation, 2) motion recovery, and 3) action recognition. Human pose estimation refers to estimating a 3D pose from a single image, while motion recovery is to reconstruct 3D human motion from consecutive frames. In this thesis, we use the term human motion recovery to refer to these two processes – 3D pose estimation, followed by motion

reconstruction. Both hand-crafted features and self-learning features from deep learning approaches are also investigated in accordance to the size of the dataset available. Since both spatial and temporal features are important in a motion sequence, we look into learning compact spatio-temporal features across consecutive frames.

1.2 Motivation

The motivation behind this research arises from three different considerations: 1) limitations in traditional motion capture, 2) useful real-life applications, and 3) high volume of readily available data.

While existing methods for motion capture, using optical markers, multiple cameras, and depth sensors, are commonly used, they are limited to constrained environments, suffer from high costs and require specific devices and manual operations. The full process is often time-consuming, labor intensive and complex in both hardware and software settings. They involve environment setup, equipment calibration, sensor registrations, followed by tracking, synchronization and noise removal after obtaining sensor data and videos from multiple cameras.

With these limitations, it is difficult to implement the system for outdoor applications such as sports analysis and surveillance tracking, and certainly not for small establishments with little human and financial resources. We seek a better alternative which allows us to perform motion recovery and recognition while overcoming all the above-mentioned constraints. Hence, our system should satisfy the following conditions:

- 1) Human motion analysis and recovery in any environment.
- 2) Does not require any special devices to capture or process the motion data.
- 3) Low cost and easy to operate.

By exploiting monocular video inputs for human motion analysis, we could capture motion data using a camera or mobile phone, anytime anywhere. Implementation for outdoor applications will also be feasible. Besides

capturing our own motion data, we could also access a high volume of rich content videos that are publicly available from the internet. However, it will be extremely labor extensive to collect and annotate the actions or 3D poses. Nevertheless, thanks to the research community in publishing motion dataset with annotated labels, we are able to investigate various processes in human motion recovery, especially in 3D pose estimation [11, 12] and action recognition [13, 14].

1.3 Scope and Objectives

The main objective of this thesis is to investigate the processes in vision-based human motion analysis, and focus on two main processes: 1) human motion recovery and 2) action recognition. Human motion recovery is a regression problem, while action recognition is a classification problem. For and the task of human motion recovery, we aim to reconstruct smooth and realistic poses from a sequence of frames, which could be useful in motion retargeting for animating 3D models in AR or VR applications. Due to the complexity in reconstructing arbitrary poses, we selectively focus on common actions such as walking or running, which have smooth and continuous motion. For action recognition, we research on developing a compact architecture that allows efficient learning of discriminative spatio-temporal features, while maintaining a practical computational cost.

To extract efficient feature representations, we study both hand-crafted features and self-learning features in the spatio-temporal dimension. Hand-crafted features require prior knowledge on the features to be extracted, such as appearance, shape or motion. Image processing may be required to remove noisy background and allow better detection and extraction of the features. Self-learning features can be obtained using deep learning approaches, typically with a Convolutional Neural Network, when a huge dataset is available. Useful underlying features can be learnt during supervised training to discriminate between different classes.

Learning-based methods allow learning of a model representation from a given dataset, utilizing strategies such as probabilistic approach, optimization or stochastic sampling. Two learning models were studied while addressing the tasks: model-based and non-model based. For model-based learning, we focus on deep learning methods that exploit artificial neural network algorithms and convolution functions. This approach has demonstrated superior potential in discovering discriminative features while solving multi-class classification problems. For the non-model-based approach, the nearest neighbor algorithm was investigated. It is an instance-based learning model which makes predictions based on the similarities between the training dataset and the query instance. It is a non-parametric model that does not require any training procedures and can be applied to both classification and regression problems.

Due to time limitation to cover both the two main tasks, this thesis does not cover collection of action-specific dataset for experiments. However, popular datasets that are publicly available are utilized to ease our evaluation and benchmarking process. Another limitation of this thesis is the resources constraint on computational power to train complex deep learning models. Hence, we design the scope of this study into investigating efficient encoding of spatio-temporal features while achieving competitive performance as complex deep learning models.

The investigations in this thesis will ultimately lead to the development of a complete system for human motion analysis of monocular video sequences that can be applied to specific applications such as healthcare and sports analysis.

1.4 Organization

The organization of this thesis is as follows:

Chapter 1 introduces the background and motivation behind this research work and stated the objectives to be addressed.

Chapter 2 reviews the literature on the topics of human pose estimation and action recognition based on different learning approaches. A summary of research gap is presented after the review, together with suggestions to further investigate promising approaches.

Chapter 3 works on resolving human pose estimation and motion recovery tasks by exploiting example-based learning with hand-crafted features. The work starts from image processing, feature extraction and encoding, computing similarities and alignment between sequences of images and finally interpolating the poses for all video frames to reconstruct smooth motion.

Chapter 4 introduces a compact network that learns spatio-temporal features for action recognition. 2D CNN is used for spatial features extraction, followed by a 3D convolution layer for spatio-temporal encoding. The architecture is structured in an end-to-end framework for efficient learning.

Chapter 5 extends the architecture from Chapter 4 to allow learning of hierarchical features representation. This chapter also investigates different feature aggregation functions for better learning of local and global motion dynamics.

Chapter 6 proposed a new architecture for efficient learning of spatio-temporal features by exploiting 1D, 2D and 3D convolution layers in a single stream network. It can be adapted to existing CNN models while preserving each network's learning properties. It outperforms traditional 2D and 3D CNNs as it is able to learn spatio-temporal features with lower complexity which prevents overfitting problems.

Chapter 7 summarizes the contributions of the chapters and concludes the thesis with future directions.

CHAPTER 2

LITERATURE REVIEW

In chapter 1, it is noted that human motion analysis from monocular video sequences requires a number of processes, such as human detection, pose estimation, motion tracking and action recognition. The main objective of this thesis is to investigate the main processes in 1) human motion recovery and 2) action recognition. An overview of the literature on these topics will be covered in this chapter. The first section reviews the approaches from traditional hand-crafted features with machine learning and deep learning for human motion recovery, while the second reviews various models that use deep learning for action recognition.

2.1 Survey on Human Motion Recovery

The survey in this section covers works on both single image pose prediction and consecutive frames motion reconstruction. The literature on 2D and 3D pose estimation is vast. A number of surveys already exist [1-3, 5, 7, 15-19], reviewing the work of 2D and 3D pose estimation from a single image, as well as the extension to motion recovery from video. For a better understanding on vision-based motion capture, readers are advised to read the comprehensive surveys of Moeslund, et al. [1, 7], covering years 1980 to 2006. They provide an overview on the functional processes, from system initialization, tracking, pose estimation to action recognition. Hen and Paramesran [2] reviewed image features and methods used in 3D pose estimation and tracking. Ji and Liu [5] analyzed view-invariant human motion analysis in the context of human detection, pose representation and estimation, followed by activity understanding. Perez-Sala, et al. [3] focused on model-based 2D and 3D pose recovery, with taxonomy on appearance, viewpoint, spatial models, temporal models, and behaviour. Liu, et al. [15]

looked into body part parsing methods in images and videos, and presented a list of benchmark dataset for 2D and 3D pose estimation. Nishani and Çiço [17] explored the performance of human pose estimation by investigating the implementation of deep learning methods and Convolutional Neural Network.

2.1.1 Taxonomies

Methods used for pose estimation depend on whether a model exists *a priori*. Moeslund and Granum [7] divided the taxonomy for pose estimation into three categories: model-based, indirect model and model-free. Model-based is defined as having a 3D human model *a priori*, mostly represented by a skeleton structure with kinematic chain. In indirect model, the model is usually represented by relevant information such as appearance, spatial and motion features, to serve as a reference for data inferencing. Model-free methods may be implemented using a bottom-up approach, tracking and labeling 2D body parts, or employing a probabilistic approach to detect and assemble parts configuration [1]. Another possible approach is the example-based, where a 3D pose is estimated from a reference model, or from a learning model of database samples.

Sminchisescu [20] classified the reconstruction of 3D human motion into two approaches: generative (top-down) and discriminative (bottom-up). Generative methods optimize the alignment of 2D image features with 3D human model, where the optimization encodes a likelihood or cost function. Discriminative methods estimate human poses directly from images using training sets that consist of joint poses and image features. This approach is also viewed as a recognition-based approach where machine learning is used extensively [2].

We categorise the taxonomy in this section based on its model representation (model-based or model-free) and the estimation approach (learning-based or example-based). Table 2.1 provides an overview of the categories of taxonomy which will be explained further in the following sub-sections.

Table 2.1 An overview of human pose estimation taxonomies.

Approach	Description
Model-based	The model is represented by an underlying skeletal structure. The estimation process optimizes similarities between estimated pose projection and the given 2D image.
Example-based	It does not require an underlying model representation. Pose prediction is obtained by finding the nearest images from the training dataset with distance comparison.
Learning-based	The algorithm learns mapping functions between a 2D image input and its corresponding 3D pose. This category covers both machine learning and deep learning methods.
Dynamic model	This model considers time dependency which estimates changes in human pose over time. It enables pose tracking and motion reconstruction.
Others	This covers multi-task learning and semi-supervised methods.

2.1.1.1 Model-based Approach

A human model can be represented by an articulated skeleton structure with bones connected at joints, constrained by individual degrees of freedom. The process can be described as measuring and optimizing the similarities between an estimated pose projection and a given 2D image. Typically, the process is repeated for a number of iterations until the best matched pose is found and used to update the model in the system [7]. Due to the high dimensional state space of human configuration and visual ambiguities from monocular images, a large number of possible 3D poses could match a 2D projection image. In order to reduce the search space and obtain a good prediction, several approaches can be applied such as imposing kinematic constraints, stochastic sampling, probabilistic method and using learnt models [1].

Felzenszwalb et al. [21] proposed to use a deformable part model (Figure 2.1) for 2D body parts detection and pose estimation. A deformable part model, which is developed from a pictorial structure model [22, 23], is

used to represent a deformable object formed by different connected parts. Each part model consists of a part filter to capture local features, and a spatial model to express the spatial relation between parts. Andriluka et al. [24] extended the pictorial structure model with strong part detectors combined with kinematic tree prior. They modeled body parts based on shape context descriptors [25] and applied the AdaBoost algorithm [26] to train individual part classifiers. Kinematic constraints on part configuration are trained from the dataset and are encoded as a flexible kinematic tree.

Figure 2.1. (a) A pictorial structure of human body [23]. (b) A deformable part model represented by a coarse global template (left), parts template (middle) and spatial model (right) [21].

Wang and Mori [27] considered a deformable part model constructed from a weighted combination of multiple tree models. Different tree models are trained to capture spatial constraints and occlusion between parts. Eichner and Ferrari [28] extended the pictorial structure model for multi-person pose estimation, integrated with occlusion priors. Sapp et al. [29] trained a cascaded pictorial structure model for progressive search space reduction using rich appearance features. The coarse-to-fine pruning process is conducted based on the max-marginal scores from the computation of dynamic programming.

Dantone et al. [30, 31] enhanced part detection in a pictorial structure by incorporating joint regressors that predict the location of joints with respect to the co-occurrence of other body parts. Joint regressors are trained with random forests which consist of two layers for prediction: the first layer

performs parts detection, while the second layer predicts joint location based on the estimated part location. Ladicky et al. [32] trained a mixture of part models using rich appearance features – Histograms of Oriented Gradients (HOG) [33], color, texture, and body part segment. The four components form the energy function for pose estimation. The method works well on images that have multi-person interactions and are highly cluttered.

Yang and Ramanan [34, 35] implemented a flexible mixture-of-parts model (Figure 2.2) for articulated human detection and 2D pose estimation. Different from traditional approaches that capture a part's orientation in the model, a flexible mixture-of-parts model approximates the parts' transformations using non-oriented image patches connected with different spring models to represent different spatial arrangements.

Figure 2.2. (a) Oriented part model in traditional approach. (b) Non-oriented image patches used to approximate limbs' transformation. (c) Various spring models are used to connect image patches for different spatial arrangements [34].

Bourdev and Malik [36] proposed a novel definition of a body part called poselet. It is defined as a part that is tightly clustered in both the appearance and configuration space. Poselets are trained as body part detectors that can be applied in 2D and 3D pose estimation. Wang et al. [37] extended the poselet representation into a hierarchical structure that consists of rigid parts, regions of body parts, and the whole body, as illustrated in Figure 2.3. The framework is represented as a graph structure where a vertex denotes a part and an edge denotes connectivity between parts. Pishchulin et al. [38] improved the tree-structured pictorial structure model using poselets as parts detector to capture inter-part dependencies and articulation.

Figure 2.3. A graph structured hierarchy pose representation. Colored rectangles denote body parts (parameterized with poselet templates) and black edges denote relation between parts [37].

Pirsiavash and Ramanan [39] introduced a novel steerable part-based model to describe large variations of part articulation by utilizing a sparse set of basis filters. The steerability and separability of the basis filter is learnt from a mixture of part dataset (across different viewpoints and object categories) using latent SVM [40] approach. Another novel representation called Posebits (Figure 2.4) was proposed by Pons-Moll et al. [41], to provide semantic description for pose retrieval and inference. It considers geometric relationships between parts that correspond to distances between joints, bending angles, and part positions. Each posebit can be trained using an SVM classifier with spatial pyramid HOG features.

Figure 2.4. Three types of posebit that consider: joints distance, bending angle, and between parts position [41].

2.1.1.2 Example-based Approach

Without the use of an *a priori* model, pose data can be estimated from the nearest neighbors in a training dataset. Similarities between a test image and training images are computed to find the best matching pose with the

shortest distance. Temporal information can also be incorporated for the estimation of the pose motion.

Shakhnarovich et al. [42] proposed an example-based method for fast pose estimation. Images are represented by multi-scale edge direction histograms, and a parameter-sensitive hashing algorithm is applied to search for the nearest neighbors. The final pose is estimated using a robust locally-weighted regression method. Poppe [43] evaluated example-based pose estimation approaches where histograms of oriented gradients are used as image descriptor. Their results show that the estimation process is less robust against variations in human appearance and unseen poses.

Fragkiadaki et al. [44] exploited motion-based segmentation for better parts detection and tracking. Segmented pose is matched with exemplars to obtain articulated joint position, which in turn improve on motion tracking by imposing kinematic constraints.

Liu et al. [45] worked on 3D human motion tracking by introducing an exemplar-based conditional particle filter method, which estimates a motion transition function from the previous pose to the current matched exemplar. Zhou and Li [46] investigated both the spatial and temporal information on human silhouette sequences to recover corrupted or occluded images. They proposed to learn the shape dictionary of a highly articulated object by representing each image as a linear combination of local shape features.

2.1.1.3 Learning-based Approach

An alternative approach to estimate 3D poses without an explicit human model is the learning-based technique. From a set of training examples with known 3D poses, mapping functions are learnt between 2D image data and the corresponding 3D pose.

Agarwal and Triggs [47] described a learning-based approach to estimate 3D poses from training examples. Image observations are represented by a histogram of shape context descriptors, while body poses are parameterized by vectors of joint angles. Hara and Chellappa [48] constructed

a dependency graph model as pose representation to capture the relationships between joint locations. The dependencies between joints are trained with a regressor, which estimates the child nodes' positions based on the parent node's location.

Simo-Serra et al. [49] proposed a stochastic sampling strategy to estimate 3D human pose from noisy projection image (Figure 2.5). 2D body parts are detected and modeled as Gaussian distributions in the image plane, before propagating to the shape space. By imposing geometric and kinematic constraints on the sampled 3D poses, an accurate final solution can be obtained.

Figure 2.5. 3D pose estimation from noisy observations. (a) Raw input image (b) Detected parts location (c) Projection of estimated 3D poses on the scene (d) Estimated 3D poses [49].

Rothrock et al. [50] presented a grammar model structured in an and-or graph to model pose articulation. The graph model is used to learn probabilistic distribution of parts based on appearance features, geometric information and compatibility between parts. Appearance features are extracted from segmented regions and HOG computation.

2.1.1.3.1 Deep learning with Convolution Neural Network

Jain et al. [51] were the first to use deep learning for features learning in human pose estimation. They utilized a multi-layer convolutional network to learn image features and spatial priors between parts as illustrated in Figure 2.6. The process starts with an RGB image patch input, and processed by convolution and subsampling layers, followed by fully connected layers and an output layer to predict a part's presence.

Figure 2.6. The architecture of convolutional network to train features for human pose estimation [51].

Li et al. [52, 53] trained a multi-task deep convolutional neural network for both detection and regression of joint locations. The architecture consists of convolution and max-pooling layers that are shared between the detection and regression network. The last layer is connected to two separate networks – detection and regression networks, each consisting of three fully connected layers. Figure 2.7 illustrate the framework of the multi-task learning network.

Figure 2.7. A multi-task deep learning architecture that jointly trains detection and regression of joint positions [53].

Tompson et al. [54] combined a convolutional network and graphical model in a unified model. The CNN is trained as a part detector, where the input image is processed in an architecture of sliding window, multi-resolution and overlapping receptive fields. The output of the part detector is a heat map that shows possibilities of joint location. To improve on the joint prediction, spatial relations between joints are modeled using a graphical model that approximates Markov Random Field inference. On the other hand, Chen and Yuille [55] used a graphical model for pose representation and

learned its pairwise parts relationship using deep convolutional neural networks.

Toshev and Szegedy [56] presented a cascaded Deep Neural Network method to regress body joint location from images. In contrast to previous methods, they require neither part detectors nor graphical model inferencing. The cascaded regressors refine the final joint location based on previous estimations to achieve a higher accuracy. Pfister et al. [57] also trained a deep network using heatmap regressors for joint location, spatial relation between joints, and temporal alignment with neighboring frames. The combined heatmaps are pooled to predict the final joint locations.

Newell et al. [58] proposed a novel stacked hourglass network for joint top-down and bottom-up processing. A single hourglass network is designed to process spatial information at different resolutions and output joint predictions in heatmaps as shown in Figure 2.8. Stacking multiple hourglasses with intermediate supervision allow refined prediction by assessing features from both the local and global context. Mehta et al. [59] demonstrated a real-time 3D human pose reconstruction using a CNN regressor for joint estimation, combined with temporal tracking and 3D skeleton fitting. The result is claimed to be comparable to reconstructed poses from RGBD images.

Figure 2.8. A stacked hourglass architecture for repeated bottom-up and top-down processing [58].

2.1.1.3.2 Deep Learning with Recurrent Neural Network

Fragkiadaki et al. [60] employed a recurrent neural network (RNN) model for tracking and predicting body joints in video frames. The RNN is implemented using the architecture of Encoder-Recurrent-Decoder, which can perform feature learning and kinematic tracking given an image sequence.

Belagiannis and Zisserman [61] also trained a recurrent CNN architecture that learns part detection and keypoint regression. The convolution layers are used to extract spatial information and feed to the recurrent network to predict heatmaps of body parts and keypoint locations.

2.1.1.4 Dynamic model

Gaussian process latent variable model [62] is a probabilistic model that learns a latent space mapping between the observation space (appearance features) and the pose space (joint angles). Incorporating Gaussian Process in the latent representation allows dynamic modelling for pose tracking [63]. Andriluka et al. [64] proposed a dynamic model that detects and tracks limb motion in consecutive frames, represented in a low dimensional latent space using a hierarchical Gaussian process latent variable model [65]. Reconstruction of 2D pose is formulated as an optimization problem that maximizes the detection likelihood, sequential pose prediction and the latent space prior. They further extended their work to include viewpoint classification and tracking to enable 3D pose estimation [66].

Ferrari et al. [67] proposed a progressive approach for 2D upper body pose tracking. The framework starts from body parts detection, parts segmentation from background, followed by image parsing using a pictorial structure model. Temporal tracking between parts is inferred using belief propagation and is integrated in a spatio-temporal model. From the estimated pose, they further defined different pose descriptors as representation for pose retrieval application [68].

Elgammal and Lee [69] introduced a supervised manifold learning method to learn a shared representation between the visual input of human shape contours and the output of motion kinematics. The paper shows that the topology of a torus manifold is suitable for both visual and kinematic embedding. The trajectory of motion can be tracked on the torus, and the pose is reconstructed using the learnt non-linear mapping functions from the torus representation to the visual and kinematic manifolds.

Bo and Sminchisescu [70] presented a twin Gaussian Process model to learn the non-linear correlations between the input and output distributions. It is achieved by minimizing the Kullback-Leibler divergence so that the estimated output is matched to the input distribution. Taylor et al. [71] used another class of latent variable model, called Conditional Restricted Boltzmann Machine [72], that is able to capture the local constraints and global dynamics for 3D motion tracking.

2.1.1.5 Others

Yao and Fei-Fei [73] trained a context-based pose estimation that jointly learns the objects and human poses in human-object interaction activities. The inference framework includes object detection, followed by pose estimation based on maximum likelihood of pre-trained body-object configuration. Eichner and Ferrari [74] presented a new perspective of estimating multiple human poses performing the same action, but with unknown poses. They addressed this problem as human pose co-estimation (PCE) and applied a multimodal hierarchical pictorial structure model for joint pose estimation. They also showed that PCE can be applied as a weakly-supervised approach to learn different pose classes from web queries.

Abbasi et al. [75] proposed a semi-supervised graph-based approach to estimate 3D poses from a small set of labelled data. A frame sequence from the same activity (labelled and unlabelled) can be used to construct a graph model by finding its k -nearest neighbours (k -NN) and temporal relationships with consecutive frames. The data are used for manifold learning such that similar images that are near to each other share the same underlying 3D pose.

2.1.2 Summary

Although significant progress has been made in human pose estimation research, automatic initialization of poses for the model-based approach still remains problematic [5]. A recommended direction is to combine both the example-based and model-based approaches for pose estimation [5]. Imposing contextual constraints and multiple feature extraction may also help

in increasing the performance of reconstruction using the example-based approach [5]. Besides, pose estimation in complex scenes such as cluttered background is not yet mature [16]. Efficient search methods and rich image features can be explored for better matching of appearance models and reduction in computational cost.

In the research of human motion recovery, there exists a large gap between theoretical approaches and real-world applications [15]. Various works exist for motion analysis in surveillance to track human movements and identify abnormal activities. However, only limited works focus on motion recovery and action recognition in sports video [76]. One of the major challenges is the segmentation of the subject from complex and dynamic background [76]. It requires additional processing for noisy data in order to recognize the segmented body.

Tracking of 3D human motion is challenging as most of the existing particle filter-based methods cannot handle motion prediction with sudden changes [45]. Temporal jittering is a common drawback that occurs when reconstructing human motion with per-frame pose estimation [59]. Imposing temporal coherence between estimated poses between frames would help to reconstruct smooth and continuous motion.

In the following chapter, we propose to exploit the example-based approach on a stack of consecutive frames for better pose initialization in sequential frames. Image pre-processing will be necessary to align the subject in consecutive frames such that temporal coherence can be obtained. Both spatial and temporal features then form an important descriptor to find the best matching exemplar in the training dataset for pose reconstruction.

2.2 Survey on Human Action Recognition

2.2.1 Taxonomies

For a sequence of video frames, typically two major features constitute the feature representation – spatial appearance and temporal motion flow.

Spatial information can be extracted using a CNN model, where hierarchical feature representations can be learnt for different action classes. Temporal features can be encoded using RNN or by applying CNN on motion features of consecutive frames. Different architectural networks have been developed and trained for the tasks of motion recognition, action segmentation and classification, learning motion feature and action video retrieval. The taxonomy used in this section is split according to the network structure and applied approach as stated in Table 2.2.

Table 2.2 Taxonomy for action recognition approaches.

Approach	Description
3D CNN	This model employs 3D convolutions to directly extract both spatial and temporal features across a stack of video frames.
Multi-stream CNN	Spatial and temporal features are captured in separate CNNs before combining them to train a classifier.
RNN/ LSTM	This model allows sequential learning, which tracks temporal features over a long period of time and is able to predict future motion.
CNN with additional information	This approach requires a pre-processing stage for body parts detection or pose estimation before action recognition.
Unsupervised learning	This category may adopt RNN/ clustering/ semi-supervised approach to learn motion features.

2.2.1.1 3D CNN / Spatio-temporal CNN

In order to capture both spatial and temporal features across video frames, Ji, et al. [77] proposed a 3D CNN (Figure 2.9) model to perform 3D convolution on multiple frames. This model achieves better performance in action recognition as compared to 2D CNN. However, the temporal information is captured in a fixed time frame and does not provide dynamic information over a long period of time. Jung, et al. [78] improved the model by capturing multiple timescales at different layers of the convolutional network.

A higher layer encodes a longer time frame than a lower layer, and the feature representation is useful for motion recognition over a long range.

Figure 2.9. The architecture of a 3D CNN [77].

Karpathy, et al. [79] designed a CNN baseline architecture and studied the performance of temporal fusion in 3 different settings: early fusion (temporal information extracted across adjacent frames), late fusion (features extracted from 2 frames that are a distance apart), and slow fusion (similar to 3D CNN). Good performance can be obtained with the baseline of a single frame feature, but extension of temporal slow fusion performs best. Furthermore, they also demonstrated that low-level features from a trained network is beneficial for transfer learning to recognize smaller sets of sports actions. It speeds up the training process, as it only requires fine-tuning at the top layers, instead of training from scratch for the new dataset.

Working with the full 3D spatio-temporal kernel requires a large training dataset and incurs high training complexity. To overcome this difficulty, Sun, et al. [80] presented a more effective way of processing the 3D signal by factorizing the 3D kernel into a sequence learning of 2D spatial kernels and 1D temporal kernels, with comparative performance as the full 3D kernel. Ng, et al. [81] trained a multi-task model, called ActionFlowNet, that learns motion features while recognizing actions.

2.2.1.2 Two-stream / Multiple-stream CNN

Instead of constructing a spatio-temporal CNN using 3D feature maps, Simonyan and Zisserman [82] designed a two-stream CNN architecture

(Figure 2.10) which captures spatial and temporal features in separate CNN before combining them to train a classifier. A spatial stream extracts features from individual video frames; while a temporal stream extracts dense motion trajectories by computing the optical flow over consecutive frames.

Figure 2.10. A two-stream CNN architecture to learn spatial and temporal feature for action recognition [82].

Wang, et al. [83] integrated CNN features and hand-crafted features to jointly learn a novel motion representation, called trajectory-pooled deep-convolutional descriptor (TDD). CNN features are learnt from a two-stream model that consists of both spatial and temporal streams. Improved dense trajectories [84] are computed from the optical flow, and is later used to pool CNN features along their trajectories, forming the TDD representation. It is shown that utilizing hand-crafted features such as dense optical flow as temporal representation performs better than trained CNN motion features from video frames [82, 83, 85].

Similar to the two-stream CNN, Gkioxari and Malik [86] trained separate CNN for video frames and flow signals. Instead of using the full image as input, their system search for region proposals and train the classifier for action prediction in individual frames. The identified action regions will be linked with consecutive frames to form connected motions, called action tubes. Tu, et al. [87] also employed human-based region proposals to train a multi-stream CNN that consists of appearance, motion and region features as input stream.

Zhang, et al. [88] achieved real-time action recognition using a two-stream CNN architecture. Instead of computing optical flow as the input for a temporal stream, they learn motion vector from a pre-trained optical flow network, which in turn reduces the processing time. Sun, et al. [89] also presented a derived motion feature from optical flow that is faster and more robust than utilizing optical flow. The optical flow guided feature can be effectively learned from adjacent frames in a single-stream network.

Diba, et al. [90] introduced an efficient and compact representation by encoding features from each CNN stream using temporal linear encoding (TLE) and concatenate them to form a descriptor. The encoding approach can also be applied to a 3D CNN architecture. Another work by Girdhar, et al. [91], learned aggregation of features for both spatial and temporal streams, to form a new encoded representation in action classification. Feichtenhofer and Zisserman [92] introduced a fusion of two-stream CNN and 3D CNN, where various fusion ways are investigated for both spatial and temporal streams. A three-stream architecture was presented by Huang, et al. [93] with an additional CNN stream on movement features, represented as motion vectors of the subject centroid. The system is proposed to be applied in home care centers for identifying dangerous actions.

2.2.1.3 Pose-based CNN

Chéron, et al. [94] presented a different scheme for action recognition by introducing pose-based CNN features (P-CNN), as shown in Figure 2.11. The procedure starts with estimating body joint positions for each frame, which utilizes the deformable part model approach. From the detected joint locations, body parts are cropped into image patches and later used to train the CNN. Two CNN streams are constructed to learn spatial features and motion flow features. Features from both streams are concatenated for all the parts to form a pose descriptor, P-CNN. This approach is more robust to pose estimation errors and achieves superior performance in fine grained action recognition.

Figure 2.11. An architecture with pose estimation framework, and body parts spatio-temporal feature extraction. Concatenation of all body parts' feature form the descriptor P-CNN [94].

Inspired by both works of pose-based CNN [94] and detecting of local motion to form action tube [86], Tu, et al. [95] proposed a descriptor based on motion salient region (MSR), which outperforms both in action recognition. MSR is estimated from a video frame by detecting foreground movement and tracking the motion consistency. Extracted motion regions from each frame form image patches to train CNN-based spatio-temporal features.

Cao, et al. [96] introduced body joint guided 3D CNN features to form an action descriptor called joints-pooled 3D deep convolutional descriptor (JDD). Body joints can be estimated using the existing pose reconstruction method, and 3D feature maps are pooled and learnt according to the joint positions. They also proposed a two-stream framework that jointly estimates body joint positions and extracts pooled spatio-temporal features. Choutas, et al. [97] proposed a novel pose motion representation that employs human pose estimator to extract heatmap of body joints and later encodes motion representation from the trajectories of keypoints.

2.2.1.4 Recurrent Neural Network / LSTM / Hybrid approach

The RNN/ LSTM (Long short-term memory) approach is often implemented as a hybrid architecture that combines with CNN features learning. Hou et al. [98] proposed a two-stage framework which utilizes 3D CNN to extract spatio-temporal features, followed by training an LSTM recurrent network [99] to classify the motion of a video sequence (Figure

2.12). A sequential architecture is proposed by Hu, et al. [100] to encode a sequence of spatio-temporal features (extracted from pre-trained CNN) using RNN with Gated Recurrent Unit (GRU) [101], and later learn the semantic meaning behind the video context. A mapping function is learnt between the sequence of video representation and semantic labelling of the context. The proposed approach not only is capable of performing action recognition, but also allows semantic video retrieval based on query words.

Figure 2.12. The framework for sequential learning structured with 3D CNN and LSTM for action classification [98].

Singh, et al. [102] introduced a multi-stream network that trains spatial and temporal features on both full frame images and region images with local motion. Features are then fed to a bi-directional LSTM to train action recognition over a long sequence of video. Utilizing bi-directional LSTM allows learning of temporal information for past actions as well as future actions. It is also able to segment a multiple-action video into different action classes temporally.

Ng, et al. [103] extended the two-stream CNN architecture to capture long range actions, where one stream performs convolutional feature pooling, while another applies LSTM to encode temporal information. Different from previous temporal networks that utilize optical flow and motion trajectories, Hou, et al. [104] replaced the learning of temporal features using RNN with a mutation of gated recurrent units, where the architecture consists of a pre-trained CNN model, followed by two separate streams of spatial and temporal networks.

2.2.1.5 Unsupervised Approach

Srivastava, et al. [105] proposed an unsupervised learning approach using RNN with LSTM model constructed in an autoencoder [106] structure. The model is not only designed to learn good feature representation from an action sequence, it is also trained to make prediction for future frames. Experimental results showed that features from unsupervised learning are useful to improve supervised classification tasks.

Misra, et al. [107] conducted unsupervised learning on unlabeled video to extract spatio-temporal features with the correct temporal sequence. The system is built with a triplet Siamese network, where the inputs are positive and negative sequences of frames from a high motion video. The network is trained to verify the correct order of temporal sequence. Leant feature representation can be applied in the nearest neighbor retrieval and serves as pre-trained initialization for supervised tasks.

2.2.2 Summary

A number of recent works in human pose estimation and action recognition is reviewed based on the adopted learning approaches. Appearance feature, temporal flow, motion trajectories are strong features to train for pose and action representations. Most of the reviewed works use the supervised learning approach that learns model representation from a given training dataset. Generally, supervised learning achieves superior performance when trained with deep learning models such as CNNs and RNNs. For unsupervised learning, it is challenging to learn discriminative features from unlabeled poses or videos.

Given a sequence of video frames, information is represented in the spatial dimension and temporal dimension. Spatial feature can be effectively extracted by training a CNN to identify action pose and localized motion. Temporal feature can be encoded using hand-crafted motion trajectory, or can be trained directly using CNN or RNN. CNN models train using fixed length videos that often could not capture the action dynamics. On the other hand,

RNN can be trained over longer periods. A hybrid system takes advantage of both networks, and is trained to identify local appearance features, as well as the dynamics of an action over a long sequence video.

In Chapters 4 to 6, we investigate and experiment with different CNN architectures for efficient spatio-temporal learning with practical computational cost. Transfer learning is adopted to take advantage of existing pre-trained models that have shown superior learning capacity and prediction accuracy. After investigating various configurations and demonstrated the effectiveness of spatio-temporal architecture, we further generalize the concept and develop an architecture that can be adapted for existing CNN models. Our model outperforms corresponding 3D CNN models with better prediction accuracy, lower computational cost and faster training.

CHAPTER 3

3D HUMAN MOTION RECOVERY FROM A SINGLE VIDEO USING DENSE SPATIO- TEMPORAL FEATURES WITH EXEMPLAR- BASED APPROACH

3.1 Introduction

In this chapter, the research focuses on 3D human motion recovery from a sequence of video images using the example-based approach. Conventional human pose tracking requires two processing stages: 1) estimating the 3D pose in a single frame, and 2) using the current estimated pose to predict the pose in the next frame. This usually involves generating a set of possible poses in the prediction state, then optimize the mapping between the projection of the predicted poses and the 2D image in the subsequent frame. The complexity of this approach becomes significant when the search space dimensionality increases. Besides, wrong prediction can easily occur when there is occlusion or projection ambiguity in the frame. The pose prediction error from a single frame will be propagated to subsequent frames, causing accumulated errors for the remaining sequence.

In contrast to the conventional approach, we propose an example-based framework for direct motion estimation in video frames, addressing three main challenges in pose estimation and motion tracking: 1) iterative prediction-updating process in single-frame pose prediction, 2) occlusion and appearance ambiguity in image frames and 3) error propagation in motion tracking.

Our system exploits coherence between a stack of consecutive frames by extracting dense appearance and motion features in the spatio-temporal space. Instead of predicting a single 3D pose at a time, our system estimates a

sequence of 3D poses for up to 12 frames, by finding the best matching frames in the training dataset utilizing dynamic time warping, ensuring smooth motion reconstruction. The system is useful for motion retargeting to animate 3D models in an AR/VR environment, action recognition and motion prediction.

3.2 Related Work

Commonly used spatio-temporal descriptors for video includes 3D volumes of Histograms of Oriented Gradients (HOG) [33], Histograms of Optical Flow (HOF) [108] and Motion Boundary Histograms (MBH) [108]. Uijlings et al. [109] evaluated the performances of these descriptors with various quantization methods in the context of video classification and proposed an efficient implementation of dense descriptors by dividing a video volume into 3D blocks. Wang et al [110] proposed a dense representation based on trajectories and extracted features HOG, HOF and MBH that aligned with the trajectories as descriptors. The computed MBH along the trajectories is used to capture and remove the camera motion in dynamic video contents. Following the work of [110], Kantorov and Laptev [111] developed efficient video descriptors using sparse motion vectors to improve the speed of feature extraction. Local descriptors HOG, HOF and MBH are extracted for a sparse set of points positioned on the motion vectors.

Tekin et al [112] exploits motion information in a video sequence to reconstruct 3D human poses by training a regression model. Their work starts with aligning the subject's motion in consecutive frames before extracting multi-scale 3D HOG features over the volume. For motion alignment, they trained CNNs to estimate the shift of a subject's position and iteratively refine it such that the subject is centered. The regression model is trained by finding the mapping function between spatio-temporal features to the 3D pose. Given a video volume of 24 to 48 frames, the 3D pose in the central frame can be

predicted using the trained model. Their work is evaluated on several benchmark datasets and shows improvement over the state-of-the-art.

Different from previous work [112] that utilized CNNs for iterative motion alignment, we implemented subject alignment by pre-processing background subtraction, followed by finding the maximum overlapping area between two consecutive foreground subjects. Our work exploits spatial and temporal descriptors and applies them to track and estimate human poses in a sequence of images.

Motion estimation with regression modeling [112] requires a stack of 24 to 48 frames as input to predict a single 3D pose. Conversely, our work focuses on direct motion estimation for up to 12 consecutive frames by utilizing the exemplar-based approach, which does not require training and yet is able to provide practical solutions. In addition, we ensure a smooth reconstructed motion by interpolating the best matching exemplars in each frame.

We evaluated our approach using the HumanEVA dataset and presented both quantitative comparisons and visual results to demonstrate the advantages of our approach over model-based and probabilistic methods.

3.3 Methodology

In view of the limitations of HOG in the spatio-temporal domain, we propose to exploit motion information by computing the optical flow of the moving subject. We aim to recover the full motion in a sequence utilizing the example-based method by storing sub-video volumes (also called video patches), represented by a set of descriptors, with their corresponding 3D poses as exemplars. A video patch may contain a sequence of the original frames, or cropped frames where a full body pose is centered and aligned. The motion in a video patch can be estimated by finding the best matching exemplars in the training dataset. To ensure smooth motion between video patches, interpolation is performed to generate in-between poses. The overall

work flow is depicted in Figure 3.1. Detailed descriptions are presented in the following sections.

Figure 3.1. Overview of the proposed 3D motion recovery in video volume using densely extracted spatio-temporal features.

3.3.1 Subject Alignment

From the original frames where the subject is moving in a circular motion, extracted spatio-temporal features may capture noise from the moving subject or background which affects the performance of features matching at the later stage. Hence, to enhance the discrimination of feature descriptor in encoding better motion coherence between frames, we pre-process the frames such that the subject is aligned and centered. Before aligning the subject in each frame, we first remove the background and then find the bounding box of the subject's silhouette. A Gaussian Mixture Model is employed to learn the probability distribution of the background pixels [11, 113]. Each Gaussian model is represented by the mean and variance of the pixel colors (r, g, b values). A pixel that does not match with the Gaussian distribution is then labelled as foreground. However, this method cannot reliably remove shadow pixels from the foreground mask due to the changes in color intensities when shadows are cast. From the observation of [114], a shadow pixel can be taken to have significant intensity changes but without much changes in its chromaticity. The chromaticity can be computed as proportions of r, g, b colors in the image as in Eq. 3.1:

$$\begin{aligned}
 rc &= r / (r + g + b); \\
 gc &= g / (r + g + b); \\
 bc &= b / (r + g + b)
 \end{aligned}
 \tag{Eq. 3.1}$$

In computing the Gaussian Mixture Model, we now learn the mean and variance of the background distribution using the chromaticity values rc, gc

and bc . An example of a background image with mean chromaticity and the resulting foreground mask is shown in Figure 3.2. One limitation of using chromaticity is that it cannot differentiate the lightness of pixels – white and gray both have the same chromaticity with no color information [115]. This explains the mislabelling of the white and gray background as foreground pixels in Figure 3.2(e). Pixels that are labelled as foreground in the RGB model but appear as background in the chromaticity model are treated as shadows and are removed from the foreground mask (Figure 3.2(f)).

Figure 3.2. Extraction of human silhouette. (a) Original RGB image. (b) Background image with mean chromaticity. (c) Original image represented by its chromaticity. (d) Foreground mask obtained from RGB background model with cast shadows. (e) Foreground mask obtained from chromaticity background model (f) Final silhouette after background subtraction, shadow removal and noise removal.

After obtaining the subject's silhouette from background subtraction, we can find the bounding box of the subject. We compare the bounding box with its previous frame to find a shifted position (in the x and y directions) that gives the maximum overlapping area between two silhouettes. We investigated three different scenarios of the input image to evaluate the efficiency of motion compensation as depicted in Figure 3.3. The subject is first aligned and centered across frames before cropping the image from the original dimension of 640×480 to 240×480 . As the subject's height changes across the frames (subject walking in front and away from the camera

position), we also scale the images to the same height before aligning and centering them in the frames with dimension 400×400 .

Figure 3.3. Average image gradients across all frames in a video volume. (a) Original silhouette image, (b) Aligned and centered subject (size 240×480) (c) Subject is scaled to the same height before alignment (size 400×400).

3.3.2 Forming Video Patches and Spatio-Temporal Blocks

We extract descriptors HOG, HOF and MBH from local video blocks and concatenate them to form the descriptor for a full video volume. The process of forming video patches and image blocks is illustrated in Figure 3.4. At first, a full video volume is split into sub-volumes, which we call video patches. A video patch contains frames from time t to $t + n$, where n denotes the number of frames to encode the motion features (we use $n = 6$ or 12). In our computation, we set the default interval between two patches as one frame. If a video volume has N frames and the interval between patches is I , the number of patches is $(N - (n+1))/I + 1$. Each frame is divided into 2D areas of pixels. By concatenating the areas in consecutive frames, we build spatio-temporal blocks for features encoding.

Figure 3.4. A video volume that contains a sequence of image frames can be split into sub-volumes. (b) A video patch from time t to $t + n$ frames. (c) Each frame is divided into areas for feature encoding.

We define a single spatio-temporal block size as $6 \times 6 \times 6$, which depicts 6×6 pixels in the spatial dimension, and 6 consecutive frames the temporal dimension. Extracted features are encoded in a single block, while concatenating a number of these blocks form feature descriptors for a video patch. We experiment on two descriptor sizes - $3 \times 3 \times 1$ and $3 \times 3 \times 2$ blocks to encode 6 and 12 consecutive frames respectively. Figure 3.5 shows an illustration of a single spatio-temporal block and the forming of a feature descriptor in a video volume.

Figure 3.5. A 3D video volume is divided into spatio-temporal blocks for feature extraction and encoding. (a) A single block with size $6 \times 6 \times 6$. (b) A descriptor represented by $3 \times 3 \times 2$ blocks. Descriptors contain overlapping blocks, where a single block is reused in multiple descriptors except for the corner blocks.

3.3.3 Feature Descriptors

HOG is an edge and gradient-based descriptor. For each pixel, the magnitude and orientation of the image gradient are computed and later quantized into a local histogram with 8 orientation bins. HOG is used to

capture image appearances, while HOF and MBH compute the optical flow between frames to encode motion information. The optical flow of a pixel is its displacement between two frames. The displacement magnitude and orientation are computed and binned into a histogram for each spatial block, to form the motion descriptor HOF. Another motion feature, MBH, treats optical flow in the vertical and horizontal direction separately, forming two separate histograms from the independent local gradients. MBH is developed to detect relative movement of the subject, suppressing constant motion from the camera and the background [108].

After extracting the features for each frame, the responses are then accumulated and aggregated in the spatial and temporal domain based on a block's size. For a descriptor with $3 \times 3 \times 2$ blocks, the responses in each block are concatenated to form a descriptor of dimension $3 \times 3 \times 2 \times 8$, where 8 denotes the number of orientation bins used for quantization.

Figure 3.6 demonstrates the flow of features extraction (HOG, HOF and MBH), encoded features in the local histogram for a spatio-temporal block, and finally forming a descriptor with size $3 \times 3 \times 2$ or $3 \times 3 \times 1$. In the experimental section, the performance of motion reconstruction based on different encoding ranges of temporal block size was investigated.

Figure 3.6. Illustration of the process flow to compute HOG, optical flow and MBH features followed by forming two descriptor representations.

3.3.4 Pose Estimation

After computing the descriptor for spatio-temporal blocks, a video patch is represented by the concatenation and normalization of all these descriptors. Each video patch in the training dataset is stored together with their corresponding 3D poses. To estimate the motion of a test video patch from time t to $t + n$ frames, we find the best matching exemplar in the dataset by comparing the absolute difference between descriptors of video patches. The test video patch and the exemplar may contain the same motion but vary in speed. To resolve this problem, we implemented dynamic time warping (DTW) [116], which uses similarity measures to find the alignment of frames between two temporal sequences, as illustrated in Figure 3.7.

Figure 3.7. An illustration of matching frames in two sequences of varying speed using Dynamic Time Warping.

Distances between frames are computed by re-using the HOG features. Frames with the shortest distance are matched, and the exemplar 3D poses are retrieved and stored as candidate poses in the test video patch. After obtaining all the candidate poses in the test sequence, the final pose is estimated by averaging all the candidate poses. Figure 3.8 presents an example in which a video patch has $t + 6$ frames and the interval between

patches is three frames. Therefore, a single frame in the video volume will be overlapped by three video patches, resulting in three candidate poses for interpolation.

Figure 3.8. An illustration of estimating the 3D pose for a single frame. From the test video patches, 3D poses from the best matching exemplar are stored as candidate poses for each frame. The final estimated pose is obtained by computing the mean pose from all candidate poses.

3.4 Experimental Setup and Results

3.4.1 Implementation

Our implementation is programmed using MATLAB, built on the code released by Uijilings, et al. [109] that extracts dense HOG, HOF and MBH in video blocks. We conducted a number of experiments to investigate the performance based on:

- 1) a single feature representation or a weighted combination of all the features
- 2) encoding the descriptor with different temporal ranges – 6 or 12 frames
- 3) interval of video patch for motion interpolation – 1 or 6 frames interval
- 4) motion compensation with cropped and aligned subject

Two datasets were used in the experiments - silhouette images and color images. The first three experiments are evaluated on the silhouette dataset to obtain good configuration settings, while the fourth is conducted on color images.

3.4.2 Evaluation on Silhouette Images

The dataset provided by Agarwal and Triggs [117] consists of silhouette images of full body walking poses. The dataset consists of 1691 training images and 418 testing images with a different walking sequence from the training set. The silhouette images are cleaned and the subject is already scaled and centered. Hence, we do not require any pre-processing on the images before feature extraction.

3.4.2.1 Experiment on different feature representation and temporal range

We extract the descriptor in a single scale with a block size of $6 \times 6 \times 6$ and compare the performance with descriptor sizes $3 \times 3 \times 1$ and $3 \times 3 \times 2$. With an input image of 128×128 pixels, the concatenated descriptor sizes for a video patch are 361×72 (for blocks $3 \times 3 \times 1$) and 361×144 (for blocks $3 \times 3 \times 2$). The accuracy of pose reconstruction is measured by the mean square error between the ground truth and the estimated joint angles in degrees. The ground truth is given in a 54-D vector, comprising 18 joints and their 3 degrees of freedom in the x, y, z directions.

Table 3.1 Mean error of joint angles (degrees) for the dataset from [117] with descriptor sizes $3 \times 3 \times 1$ and $3 \times 3 \times 2$. The weighted combinations for the four descriptors are given in the first four columns.

HOG	HOF	MBHy	MBHx	Mean error ($3 \times 3 \times 1$)	Mean error ($3 \times 3 \times 2$)
1	0	0	0	4.47	3.97
0	1	0	0	4.15	3.87
0	0	1	0	4.28	3.97
0	0	0	1	4.32	4.02
0	0	0.5	0.5	4.26	3.97
0.2	0.5	0.15	0.15	4.20	3.92
0	0.7	0.15	0.15	4.16	3.90

Table 3.1 presents the results for individual descriptors – HOG, HOF, MBH in the vertical direction (MBHy) and MBH in the horizontal direction

(MBHx), as well as their weighted combination. The sum of all the weights across the four descriptors is 1.0; a single weight of 1.0 denotes that a single descriptor is used. Overall, the HOF descriptor performs better than the rest, as motion information is more distinguished in a silhouette image than the appearance information. A descriptor of size $3 \times 3 \times 2$ is able to capture more motion features giving more promising results, with improved mean error of 0.32 degrees in average when compared with descriptor size of $3 \times 3 \times 1$.

To investigate the influence of video patch interval in both the training and test video volume, we compute four sets of results using interval = 1 frame and interval = 6 frames. The results are plotted and displayed in Figure 3.9.

Figure 3.9. Comparison of results for video patch interval of one frame and 6 frames in the training and testing dataset. Table 3.1's results are plotted in the top-most chart.

Comparing the results in Figure 3.9, it is evident that descriptors with temporal block size of 2 outperform those with temporal block size of 1. The interval for test video patches of one frame (top row of Figure 3.9) works better as it estimates the mean pose from more candidate poses, whereas the interval of 6 frames for both the training and test datasets gives the worst results.

Our average error over the full video sequence is 3.89° , whereas the baseline method [117] that implemented a regressor method, obtain an average error of 6.0° . A set of examples of reconstructed 3D poses for silhouette images is shown in Figure 3.10. Figure 3.11 displays the tracking results for the torso and the left hip angle for the full sequence.

Figure 3.10. Examples of reconstructed 3D poses for selected silhouette images from the dataset provided by Agarwal and Triggs [117].

Figure 3.11. Tracking results for torso (top) and left hip angle (bottom) of the human in the dataset provided by [117].

3.4.3 Evaluation on Color Images

In this section, we perform experiments on The HumanEva dataset [11], which is commonly used for evaluating human pose estimation as they provide training and validation datasets of video sequences with synchronized 3D body poses. We utilize the HumanEva-I dataset of walking motion captured from 3 different subjects (S1, S2 and S3) at a fixed viewpoint from camera C1. It consists a total of 1529 images for training and 1403 images for testing. The evaluation experiments conducted on walking sequence will be adopted to another two actions in the HumanEva-I dataset - jogging and boxing, where the results are to be presented in Section 3.5.

3.4.3.1 Experiment on Subject's Alignment and Weighted Descriptors

In this experiment, we compare the performance of our method on 3 different input images: 1) original image, 2) subject is cropped and centered, and 3) subject is scaled to the same height and center aligned. The block size used is 12×12 pixels with 6 frames, and the descriptor contains $3 \times 3 \times 2$ of these blocks. The interval of video patches is set at one frame. For the first set of data, the original image size is 640×480 , which gives a concatenated descriptor size of 1938×144 in a single video patch. For the second set of data, the size of the cropped image is 240×480 , forming a descriptor of dimension 684×144 . As the third set of data involves scaling of the subject in the scene, this causes significant changes in the background. To focus solely on human motion, we remove the dynamically changing background before feature extraction. The output is a scaled silhouette image with the size of 400×400 , where the size of a video patch descriptor is 961×144 . A sample image for each set of frame inputs is displayed in Figure 3.12.

Figure 3.12. Sample frames for evaluation. a) Original image (640×480). (b) Aligned and centered subject (240×480). (c) Scaled and aligned subject with background removed (400×400).

The validation results for the walking action of Subject 2 are presented in Figure 3.13 to illustrate the performance comparison of different input images. Similar to the first experiment, different combinations of feature representation are also evaluated on color input. The error metric used is the mean angular error between the joint angles in the reconstructed pose and the ground truth pose.

Figure 3.13. Comparison of results on HumanEva-I dataset for Subject 2's walking action.

The results show that images with the scaled and aligned subject and background removed give the best outcome, followed by images with the aligned subject (without scaling) and then the original image. Generally, HOG gives a very high error rate as compared to the other descriptors, except when the background is removed. HOF gives high accuracy, followed by the combination of HOF and MBH. From this experiment, we observed that aligning the subject across frames is important such that the motion of body

parts between frames can be captured accurately. Moving background in the video causes noisy motion features that affect the performance of feature matching to find the best exemplars. Figure 3.14 shows the tracking results for Subject S2's torso and left hip angle in the test sequence. Examples of the best matching exemplar for each data set are shown in Figure 3.15.

Figure 3.14. Tracking results for Subject S2 in walking action in the scaled and aligned dataset, using a weighted descriptor of $0.2 \cdot \text{HOG} + 0.5 \cdot \text{HOF} + 0.3 \cdot \text{MBH}$. (Top) Torso rotation angle (Bottom) Left hip rotation angle.

Figure 3.15. Examples of best matching exemplars for video patches of three different input images. (a) Original image, (b) aligned and centered subject, (c) scaled and center aligned subject.

3.5 Comparison with State-of-the-art

We present a comparison of 3D quantitative results for the HumanEva-I dataset walking action in Table 3.2, for all 3 subjects – S1, S2 and S3. The error metric used is the mean absolute difference between 3D joint positions. We utilize the limbs’ parameters and torso’s 3D positions provided in the dataset to generate 14 joint positions – left and right shoulders, elbows, wrists, hips, knees, ankles, neck and pelvis. The results shown in Table 3.2 are not a direct comparison as different methods utilize different validation sequences and 3D model representations, such as 12-joints, 14-joints, 15-joints or 20-joints. The results for two other actions – jogging and boxing, are presented in Table 3.3. As most of the works only provide results on the walking action only, we focus our comparison on this action, while the results for jogging and boxing are presented as reference only.

Our work is compared with other pose estimation and tracking approaches that employed (i) optimization on a set of sparse 3D poses [118], (ii) inference in a graphical model [119], (iii) a Bayesian framework for joint 2D and 3D models [120], (iv) probabilistic latent variable model [71], (v) example-based method with HOG descriptor [43], (vi) deep convolutional neural network [121], (vii) regression from spatio-temporal volume [112] and (viii) structured prediction method with twin Gaussian processes [70]. We presented two results for comparison: 1) average prediction of all the 7 descriptors combination and 2) prediction from a single weighted descriptor of $0.2 \cdot \text{HOG} + 0.5 \cdot \text{HOF} + 0.3 \cdot \text{MBH}$. The results in Table 3.2 show that our method out-performs existing pose estimation and motion tracking methods while supervised learning with complex models such as CNN, regression model and dynamic model performs better than our method. However, our model does not require training and can perform direct pose estimation for up to 12 frames, as oppose to the existing implementation of per-frame estimation. As the comparisons are from different model representations, that may have caused the differences in the results. We present some examples of reconstructed 3D poses in Figure 3.16, where scaled and aligned subjects are used as input. Our fully reconstructed motion for RGB inputs and silhouette frames is available online and can be found here: <https://drive.google.com/open?id=13RtoH3H4b-umVd-JZ1Fm0u3NrCnhQIcp>.

Table 3.2 Quantitative result comparisons by computing the mean 3D absolute errors between estimated and ground truth joint positions (in mm). Our method shows the average error obtained from the individual and combined descriptors of HOG, HOF and MBH.

Author	Approach	Walking			
		S1	S2	S3	Average
Wang et al. [118]	Optimization	71.9	75.7	85.3	77.6
Sigal et al. [119]	Graphical model	66.0	69.0	-	67.5
Simo-Serra et al. [120]	Bayesian model	65.1	48.6	73.5	62.4
Taylor et al. [71]	Probabilistic model	54.3	69.3	43.4	55.7
Poppe [43]	Example-based	41.3	39.6	55.3	45.4
Our method (average over all descriptors)	Example-based	44.5	27.0	54.9	42.1
Our method (weighted combination descriptor)	Example-based	40.3	26.8	52.3	39.8
Zhou et al. [121]	Deep CNN	34.3	32.4	49.3	38.4
Tekin et al. [112]	Regression model	37.5	25.1	49.2	37.3
Bo et al. [70]	Structured prediction	38.2	32.8	40.2	37.1

Table 3.3 Quantitative result for jogging and boxing actions. Our method shows the average error obtained from the individual and combined descriptors of HOG, HOF and MBH.

Our method	Jogging				Boxing			
	S1	S2	S3	Avg	S1	S2	S3	Avg
Average over all descriptors	55.6	62.6	42.7	53.6	56.5	66.2	69.1	63.9
A weighted combination descriptor	54.8	62.6	43.0	53.5	57.2	65.9	68.8	64.0

Figure 3.16. Examples of reconstructed 3D pose for subject S1, S2 and S3 for walking, jogging and boxing actions. Images on the left shows input frame with projected 2D pose estimation, while images on the right shows 3D pose estimation viewing from a different view point.

3.6 Conclusion

This chapter investigated a robust exemplar-based model for efficient pose estimation and motion recovery for up to 12 consecutive frames. Both appearance and motion features are extracted and encoded in dense 3D blocks to represent a video patch. Our approach is able to predict the correct poses for occluded and ambiguous frames as we perform sequential estimation instead of single frame estimation. Besides, it also prevents error propagation which happens in stochastic sampling methods that fail to recover tracking errors [112].

Without any supervised training, our example-based method outperforms most of the existing tracking methods and is comparable to the state-of-the-art methods that employ deep learning. The method can be

generalized to other actions besides walking, and the implementation is straightforward and practical for small datasets. It will be extremely useful for person-specific and action-specific motion reconstruction and analysis, such as patient's rehabilitation progress, or sports coaching, where the dataset is scarce. In addition, our method ensures realistic pose estimation and smooth motion interpolation across frames. This would help in animating 3D characters where users may look for the desired motion in a 2D video and reconstruct its 3D motion to apply to the characters. For future work, we could detect and track body parts' trajectory for motion compensation instead of performing image processing and background subtraction. We could also reduce the descriptor dimension from a dense block size to a sparse part-based descriptor to reduce the time complexity in the descriptor matching and nearest neighbor retrieval.

CHAPTER 4

A SINGLE-STREAM SPATIO-TEMPORAL NETWORK FOR VIDEO-LEVEL ACTION RECOGNITION

4.1 Introduction

From this chapter onwards, we will investigate action recognition via deep learning, specifically utilizing 2D and 3D CNNs. This chapter starts with studying spatio-temporal architectures and weighted dynamic features for video-level action recognition. We aim to develop a compact and flexible architecture with the capacity to perform spatio-temporal encodings and dynamics modeling over a full video with only RGB inputs.

We propose to exploit a single stream network, which adopts transfer learning from pretrained 2D CNNs, and extend its capabilities to encode spatio-temporal features with 3D convolutional layers. We employ a sparse sampling strategy over the full video to learn local temporal dynamics, followed by forming weighted representations to enhance action-specific discrimination. Empirical studies were conducted on the architecture search, weighted feature representations and temporal encodings to demonstrate the learning capacity and efficiency of our network. With a much lower model complexity, our network is able to achieve comparable performance to the state-of-the-art 3D CNNs on the UCF-101 dataset.

4.2 Related Work

Some of the earlier works [82, 122, 123] model 2D or 3D CNNs to process stacks of consecutive frames for recognizing local motions. Video-level prediction is obtained by extracting non-overlapping frames over the whole video and calculating the average score over all the snippets prediction. This

approach is mostly used in multi-stream architectures and 3D CNNs, where the architectures are constrained to process only fixed dimension input data. The main drawback of this dense sampling strategy is the lack of global dynamics representation of the full action. Some other works proposed an over-sampling strategy [124] to capture long-term dynamics, which results in expensive computation and a higher chance of capturing noises in the video. On the other hand, sparse sampling strategy over the full video [125] has shown high capacity in learning temporal dynamics even with a lower number of non-consecutive frames. However, we may expect lower prediction accuracy when compared with full video encoding. Other approaches with temporal encoding of multiple frames or full videos provide higher level representations for motion dynamics, which leads to better prediction but at the cost of expensive computation.

4.2.1 Dynamic Temporal Encoding

For CNN training, the input dimension is often fixed at a small number of frames, such as 16 or 32. It limits the capability to capture long range features, or the dynamics of the action sequence. To address this shortcoming, various works looked into extracting long-range dynamic features by utilizing key-frames sampling, video segmentation, hierarchical representation, features pooling, or attention mechanism.

A hierarchical temporal encoding was introduced by Li, et al. [126], utilizing 3D CNNs to learn short-term motion features, followed by feature encoding to form medium-range and long-range representations. Diba, et al. [90] applied a two-stream CNN on multiple segments in a video, and learned a temporal linear encoding to form a compact representation of the full video. Another work by Girdhar, et al. [91] learned aggregation of local features for both spatial and temporal streams, and constructed a new descriptor for classification training. Bilen, et al. [127] introduced a dynamic image network that is capable of encoding a full video sequence into a single RGB image by applying a rank pooling layer in their CNN architecture. The network learns in

an end-to-end manner while capturing efficient dynamic features. Wang, et al. [125] performed sampling of snippets from the whole video to train a two-stream CNN. The prediction for each snippet is fused to form the final score for an action video. Zhu, et al. [128] trained a hierarchical temporal representation by performing temporal pyramid pooling over spatial and temporal video features.

Inspired by the efficiency of video-level sparse sampling in Temporal Segment Networks [125], we propose to develop a temporal modeling network that learns discriminative features representation from local and global dynamics for action recognition. Instead of using the multi-stream approach to encode spatial and temporal features from consecutive frames, our model only utilizes a single stream, by replacing a global 2D pooling layer with 3D convolution and pooling layers in the architecture. Besides, we also propose a weighted aggregation of local dynamic features which offers better discrimination in segmental action classification. The architecture is structured in an end-to-end framework for efficient learning of spatio-temporal features and feature aggregation.

We studied and evaluated our proposed architecture on a popular action recognition dataset - UCF101 [13], and conducted experiments on: 1) architecture search for 3D convolutional filters and pooling layers, 2) weighted feature aggregation on different pretrained models, and 3) varying range of temporal modeling. We report comparable performance with 3D CNNs in action recognition tasks utilizing only RGB frames as input.

4.3 Proposed Framework

4.3.1 Model Architecture

Our model is constructed using a single-stream approach which only takes RGB frames as input. A training video is divided into several segments, where three consecutive frames are randomly selected from each segment as the model input. With as little as three consecutive frames per segment, we

can maintain the compactness of our architecture while still able to capture its high-level spatio-temporal features. The stride size of consecutive frames can be varied to allow flexibility in encoding different range of temporal dynamics, without affecting the architecture configurations. The random selection of frames during training ensures robust prediction while suppressing noise. Figure 4.1 illustrates the underlying process in our single-stream spatio-temporal network that captures local dynamics from three video segments (early, middle and end of the action sequence).

We utilized pretrained 2D CNNs which have shown promising results on large-scale image classification, to extract intermediate spatial features for each frame. Consecutive spatial frames are then concatenated and reshaped to form 3D features with the dimension of $(temporal \times height \times width)$. These features are later convolved with 3D kernels to allow encoding of motion coherence between frames. The visualization of spatial features and spatio-temporal features are shown in Figure 4.1. When compared with spatial representation, spatio-temporal features have more complicated patterns as they encode information from both the spatial object and its local motion.

A 3D pooling layer is later added on top of the architecture to downsample the features to a 1-D representation. Figure 4.1 displays the spatio-temporal pooling features in a matrix of three rows, where each row represents features from a segment. As reviewed in the previous section, to enhance the discrimination of local features, an aggregation function is applied to compute a weighted representation of the same dimension size. Finally, the representation of each segment is passed to a fully connected layer for action prediction. The scores from all the video segments are then fused for the final video-level action recognition.

Figure 4.1. The process of our single-stream architecture that captures spatial and temporal features with weighted aggregation for video-level action recognition..

4.3.2 Features Extraction

To perform video-level prediction, we divide a given video, V , into N equal segments $(S_1, S_2, \dots, S_N)$. From each segment S_n , we randomly sample a snippet of consecutive frames $(f_1, f_{1+r}, f_{1+2r}, \dots, f_T)$, with a temporal stride of r . All segmental frames share the same network parameters in training.

Notation. The dimension of an image frame f_t for pretrained network input is set as 224×224 , which is the same as ImageNet [129] input for network

training. The output size of intermediate spatial features is $(c \times h \times w)$, where c denotes the number of output channels, h and w are the height and width of the features after convolution and pooling operations. The input for a 3D convolution layer is reshaped to the size of $(c \times d \times h \times w)$, where d represents the temporal depth, or the number of consecutive frames sampled at each segment. The kernel size for the 3D convolution and pooling kernel is defined as $(d \times h \times w)$, and the output size after global pooling will be a 1D vector of size $(k \times 1)$, where k denotes the number of filters used.

4.3.2.1 Spatial Features from Pretrained Model

We adopted pretrained models that have shown promising classification performance on ImageNet [129] to initialize our network for high level spatial features extraction. A typical structure of CNNs consists of a stack of convolutional and pooling layers, followed by fully connected layers for classification prediction. To extract intermediate spatial features from a pretrained 2D CNN, we remove the fully-connected layer and the global average pooling layer from the model.

The output dimension for the intermediate features will be $(c \times m \times m)$ after multiple layers of convolution and pooling operations, where c is the number of convolutional filters and $m \times m$ is the feature map dimension. For instance, the output features extracted from pretrained ResNet-18 [130] will have the size of $(512 \times 7 \times 7)$.

4.3.2.2 3D Spatio-temporal Features

From each video segment, we extracted three consecutive frames for encoding local spatio-temporal features. Different from other architectures' design that takes varying temporal depths (different number of frames) as network input, we vary the temporal encoding using different stride sizes (fixed input frames). Short-term local motion is encoded using a stride size of 1 to extract consecutive frames $(f_{t+1}, f_{t+2}, f_{t+3})$. Medium- or longer-range motions can be extracted with a higher stride size. For instance, stride 3 will extract frames in the range of $(f_{t+1}, f_{t+4}, f_{t+7})$. By fixing the temporal depth and

varying the stride steps, our network has high flexibility to be adopted for different ranges of local motion without re-configuring the 3D convolutional layer and kernel filters. One can also define longer temporal depth such as 5 or 7 consecutive frames and construct a multi-layer 3D convolution-pooling operation to better encode hierarchical spatio-temporal features. To retain the simplicity of our framework for preliminary study, this work only considers temporal length of size 3.

After obtaining spatial output from a single frame with dimension ($c \times 7 \times 7$), we concatenate the features from 3 consecutive frames and reshape them into the size of ($c \times 3 \times 7 \times 7$), to serve as the input for our 3D convolution layer. In our framework, we only utilize a single 3D convolutional layer, followed by 3D batch normalization [131], ReLU (rectified linear unit), and a global max pooling layer.

4.3.3 Weighted Features Aggregation

After the process of spatio-temporal encoding, the output features are connected to a fully-connected layer for classification training. A dropout layer is also added to improve training efficiency. The final video-level prediction is obtained by averaging the prediction scores from all the segments.

The above-mentioned classification prediction has one shortcoming - it only considers local dynamics for action classification. This will result in inaccurate prediction if two different actions share the same local motion in the video segments, and further mislead the final prediction when computing the average scores. As opposed to previous work [125] that explores aggregation functions for the local scores using mean, max or weighted mean, we perform the aggregation on the features representation before computing the prediction scores.

We denote the local spatio-temporal features for each segment as $\mathcal{G}(S_n)$, where $\mathcal{G}(\cdot)$ represents the composite function of 2D and 3D convolution-pooling operations. We estimate a global representation, $\bar{\mathcal{G}}$, by computing the mean of all the segments as described in Eq. 4.1.

$$\bar{\mathcal{G}} = \frac{1}{N} \sum_{n=1}^N \mathcal{G}(S_n) \quad (\text{Eq. 4.1})$$

Then, we compute a weighted aggregated representation by performing element-wise multiplication of the local features and the global mean, as shown in Eq. 4.2.

$$V_n = \bar{\mathcal{G}} * G(S_n) \quad (\text{Eq. 4.2})$$

The intention of the weighted representation (as illustrated in Figure 4.2) is to preserve the local dynamics, while taking into account the global context of an action. When two different actions share similar local motions, each local feature will be weighted differently as they do not share the same global mean. It should be noted that this representation is different from weighted average or normalized weight, which divides the local features from the sum of all the features. A simple visualization of how the feature aggregation is represented is shown in Figure 4.2.

Besides improving local prediction scores, the advantage of this feature aggregation is that the computation does not impose any additional parameter learning in our network. Both the global representation as well as the local aggregation features can be effectively learned with our end-to-end training.

Figure 4.2. Local features from each video segment are used to compute a global representation, followed by element-wise multiplication to obtain an aggregated feature representation.

4.3.4 Architecture Comparison

Our architecture shares a similar structure as the spatial network that utilizes 2D CNN for action recognition as in [125]. Figure 4.3 compares the difference in architecture design for a spatial network from [125], a spatio-temporal network (ours) and a weighted spatio-temporal network (ours).

The architecture flow of a spatial network (Figure 4.3(a)) starts from taking an RGB frame input, extracts and encodes 2D feature representations, followed by a classification layer (depicted as a fully-connected layer) for local prediction, and finally compute the video-level prediction by averaging the local scorings.

Our spatio-temporal network (Figure 4.3(b)) replaces the 2D global pooling layer in the spatial network with a single 3D convolution and pooling layer. To enhance discriminative feature learning that takes into account local and global dynamics, we estimated a global mean and applied a weighted aggregation function to the local features, as displayed in Figure 4.3(c). The scoring for each local segment is calculated based on the weighted representations before averaging them for the final prediction.

Figure 4.3. Comparison of three single-stream architectures with RGB input: (a) spatial network [125], (b) spatio-temporal network (ours) and (c) weighted spatio-temporal network (ours).

4.4 Experiments

4.4.1 Implementation Details

The experiments in this section are conducted on split-1 of the UCF-101 dataset, which contains 9537 training videos and 3783 validation videos, encompasses 101 action classes. We perform multi-scale cropping on the images, and randomly flip in the horizontal direction to allow data augmentation. The input is then normalized using the mean and standard deviation of the ImageNet dataset. Following the implementation practice in [125], our network is trained end-to-end using stochastic gradient descent with a learning rate of 0.001, momentum 0.9, dampening 0.9, and weight decay 0.0005. The learning rate will be reduced by a factor of 10 after training for every 30 epochs. The network architecture and training is implemented in PyTorch [132] with CUDA, utilizing a single GPU NVidia Tesla K40. Due to memory constraints, we implemented accumulated gradients during training to allow processing of large batch sizes, while retaining the size of the computation graph. We use a mini batch size of 8 and accumulated the gradients for 32 iterations (the effective batch size is 256) before back-propagation.

4.4.2 Architecture Search

In the architecture search, we use pretrained Inception model with batch normalization (BN-Inception) [131] as our base model for spatial feature extraction. We experimented on the configurations for 3D convolution and pooling layers, by setting different numbers of filters and filter dimensions. The input video is divided into 3 segments, and 3 consecutive frames of stride 1 are extracted for training and validation. The architecture is trained for 60 epochs and we report the best validation results for comparison.

We study the performance of three different kernel sizes ($d \times s \times s$) for our 3D convolution-pooling layer. We set the temporal depth d as 3 and vary the spatial dimension s in three sizes – $(3 \times 1 \times 1)$, $(3 \times 3 \times 3)$ and $(3 \times 5 \times 5)$.

We also explore the number of filters k in the incremental search from 256, 512 to 1024. Each 3D convolution operation is followed by 3D batch normalization and ReLU operations, before the global pooling layer. We apply 3D max pooling with the filter sizes of $(1 \times 7 \times 7)$, $(1 \times 5 \times 5)$ and $(1 \times 3 \times 3)$ respectively, such that the output features will be reduced to the dimension of $(k \times 1 \times 1 \times 1)$, and later reshaped into a 1D output.

The configurations for the three cases are presented in Table 4.1, and we name the three settings as $k1$, $k3$ and $k5$ to indicate the difference in spatial size filters. The validation results are presented in Table 4.2 for comparison. The training and validation performances for the three cases are also displayed in Figure 4.4. The slope in the figures denotes the learning efficiency of the network, where steeper slope means faster training/ validation. Convergence is reached when the line becomes horizontal.

Table 4.1 Configurations of architecture search with different kernel sizes and dimensions for 3D convolution and pooling layer.

Layer	$k1$	$k3$	$k5$
3D Convolution	$(3 \times 1 \times 1) \times \begin{matrix} 256 \\ 512 \\ 1024 \end{matrix}$	$(3 \times 3 \times 3) \times \begin{matrix} 256 \\ 512 \\ 1024 \end{matrix}$	$(3 \times 5 \times 5) \times \begin{matrix} 256 \\ 512 \\ 1024 \end{matrix}$
3D Pooling	$(1 \times 7 \times 7)$ max pool	$(1 \times 5 \times 5)$ max pool	$(1 \times 3 \times 3)$ max pool
Classification layer	Dropout 0.8, 101-D fully-connected layer		

Table 4.2 Validation results with top-1 accuracy and number of learnable weights for different configurations. $k1$, $k2$ and $k3$ denotes three kernel sizes - $(3 \times 1 \times 1)$, $(3 \times 3 \times 3)$ and $(3 \times 5 \times 5)$. Bold value shows the best performance.

Kernel size	$k1$		$k3$		$k5$	
	# Params	Accuracy (%)	# Params	Accuracy (%)	# Params	Accuracy (%)
256	11.1M	72.16	17.4M	67.04	30.0M	53.50
512	11.9M	77.93	24.5M	75.71	49.6M	70.68
1024	13.5M	81.34	38.7M	80.54	89.0M	77.48

Configuration	Loss																								
$k1 \times 1024$	<p>Training vs validation loss</p> <p>This line graph shows the training and validation loss for the $k1 \times 1024$ configuration over 60 epochs. The y-axis represents the loss, ranging from 0.5 to 4.5. The x-axis represents the epoch, ranging from 0 to 60. The training loss (blue line) starts at approximately 4.6 and decreases steadily to about 0.6. The validation loss (orange line) starts at approximately 4.5 and decreases to about 0.7. Both losses show a slight increase after epoch 30, but remain relatively stable.</p> <table><tr><th>Epoch</th><th>train_loss</th><th>val_loss</th></tr><tr><td>0</td><td>4.6</td><td>4.5</td></tr><tr><td>10</td><td>2.5</td><td>1.8</td></tr><tr><td>20</td><td>1.2</td><td>0.9</td></tr><tr><td>30</td><td>0.8</td><td>0.7</td></tr><tr><td>40</td><td>0.6</td><td>0.7</td></tr><tr><td>50</td><td>0.6</td><td>0.7</td></tr><tr><td>60</td><td>0.6</td><td>0.7</td></tr></table>	Epoch	train_loss	val_loss	0	4.6	4.5	10	2.5	1.8	20	1.2	0.9	30	0.8	0.7	40	0.6	0.7	50	0.6	0.7	60	0.6	0.7
Epoch	train_loss	val_loss																							
0	4.6	4.5																							
10	2.5	1.8																							
20	1.2	0.9																							
30	0.8	0.7																							
40	0.6	0.7																							
50	0.6	0.7																							
60	0.6	0.7																							
$k3 \times 1024$	<p>Training vs validation loss</p> <p>This line graph shows the training and validation loss for the $k3 \times 1024$ configuration over 60 epochs. The y-axis represents the loss, ranging from 1.0 to 4.5. The x-axis represents the epoch, ranging from 0 to 60. The training loss (blue line) starts at approximately 4.6 and decreases steadily to about 0.8. The validation loss (orange line) starts at approximately 4.5 and decreases to about 0.8. Both losses show a slight increase after epoch 30, but remain relatively stable.</p> <table><tr><th>Epoch</th><th>train_loss</th><th>val_loss</th></tr><tr><td>0</td><td>4.6</td><td>4.5</td></tr><tr><td>10</td><td>2.5</td><td>1.8</td></tr><tr><td>20</td><td>1.2</td><td>0.9</td></tr><tr><td>30</td><td>0.8</td><td>0.8</td></tr><tr><td>40</td><td>0.8</td><td>0.8</td></tr><tr><td>50</td><td>0.8</td><td>0.8</td></tr><tr><td>60</td><td>0.8</td><td>0.8</td></tr></table>	Epoch	train_loss	val_loss	0	4.6	4.5	10	2.5	1.8	20	1.2	0.9	30	0.8	0.8	40	0.8	0.8	50	0.8	0.8	60	0.8	0.8
Epoch	train_loss	val_loss																							
0	4.6	4.5																							
10	2.5	1.8																							
20	1.2	0.9																							
30	0.8	0.8																							
40	0.8	0.8																							
50	0.8	0.8																							
60	0.8	0.8																							
$k5 \times 1024$	<p>Training vs validation loss</p> <p>This line graph shows the training and validation loss for the $k5 \times 1024$ configuration over 50 epochs. The y-axis represents the loss, ranging from 1.0 to 4.5. The x-axis represents the epoch, ranging from 0 to 50. The training loss (blue line) starts at approximately 4.6 and decreases steadily to about 1.0. The validation loss (orange line) starts at approximately 4.5 and decreases to about 1.0. Both losses show a slight increase after epoch 30, but remain relatively stable.</p> <table><tr><th>Epoch</th><th>train_loss</th><th>val_loss</th></tr><tr><td>0</td><td>4.6</td><td>4.5</td></tr><tr><td>10</td><td>2.5</td><td>1.8</td></tr><tr><td>20</td><td>1.2</td><td>0.9</td></tr><tr><td>30</td><td>1.0</td><td>1.0</td></tr><tr><td>40</td><td>1.0</td><td>1.0</td></tr><tr><td>50</td><td>1.0</td><td>1.0</td></tr></table>	Epoch	train_loss	val_loss	0	4.6	4.5	10	2.5	1.8	20	1.2	0.9	30	1.0	1.0	40	1.0	1.0	50	1.0	1.0			
Epoch	train_loss	val_loss																							
0	4.6	4.5																							
10	2.5	1.8																							
20	1.2	0.9																							
30	1.0	1.0																							
40	1.0	1.0																							
50	1.0	1.0

Figure 4.4. Loss and accuracy performance comparison for training (blue line) and validation (orange line) with three different kernel sizes in 3D convolution layer.

The validation results show that the recognition performance improves when we apply 3D convolution kernels with a smaller spatial size. From Figure 4.4, it is noticeable that 3D convolution with $(3 \times 1 \times 1)$ kernels learns much faster (has steeper slope) as compared to kernels with higher spatial sizes such as $(3 \times 5 \times 5)$. Increasing the number of filters also improves prediction accuracies, as it has a higher capacity to encode discriminative spatio-temporal features.

4.4.3 Weighted Features Representation

In this section, we present our implementation of the framework using different pretrained networks – ResNet [130], DenseNet [133] and BN-Inception, and compare the performances of aggregated spatio-temporal features. We only exploited models with a low number of training parameters for comparison. All the models were implemented using 1024 3D convolution filters $(3 \times 1 \times 1)$, followed by global max pooling $(1 \times 5 \times 5)$.

We report the classification accuracies in Table 4.3 and display the training and validation performance in Figure 4.5. Utilizing aggregated features before classification shows apparent improvement over all the pretrained models. An average of 7.7% improvement in accuracy can be achieved from the models in Table 4.3. As can be seen in Figure 4.5, utilizing aggregated spatio-temporal features also leads to more effective training and faster convergence.

Table 4.3 Comparison of validation results for different pretrained models that utilize local and weighted features.

Model	# Params	Local features		Weighted features	
		Top-1 acc (%)	Top-5 acc (%)	Top-1 acc (%)	Top-5 acc (%)
Resnet-18	12.9M	76.50	94.13	82.16	95.82
Resnet-34	23.0M	79.57	95.43	83.82	96.56
Densenet-121	10.2M	72.40	92.68	82.66	96.03
Densenet-169	17.7M	62.75	89.21	80.57	96.01
BN-Inception	13.5M	81.34	95.96	82.08	95.72

Model	Local features																								
Resnet-34	<p>resnet34_1024_k1_max_train_vs_val_accuracy (top-1)</p> <table><caption>Approximate data for Resnet-34 accuracy</caption><tr><th>Epoch</th><th>train_accuracy</th><th>val_accuracy</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>10</td><td>35</td><td>55</td></tr><tr><td>20</td><td>60</td><td>70</td></tr><tr><td>30</td><td>75</td><td>78</td></tr><tr><td>40</td><td>77</td><td>79</td></tr><tr><td>50</td><td>77</td><td>79</td></tr><tr><td>60</td><td>78</td><td>80</td></tr></table>	Epoch	train_accuracy	val_accuracy	0	0	0	10	35	55	20	60	70	30	75	78	40	77	79	50	77	79	60	78	80
Epoch	train_accuracy	val_accuracy																							
0	0	0																							
10	35	55																							
20	60	70																							
30	75	78																							
40	77	79																							
50	77	79																							
60	78	80																							
Densenet-169	<p>densenet169_1024_k1_max_train_vs_val_accuracy (top-1)</p> <table><caption>Approximate data for Densenet-169 accuracy</caption><tr><th>Epoch</th><th>train_accuracy</th><th>val_accuracy</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>10</td><td>10</td><td>20</td></tr><tr><td>20</td><td>25</td><td>40</td></tr><tr><td>30</td><td>45</td><td>55</td></tr><tr><td>40</td><td>52</td><td>58</td></tr><tr><td>50</td><td>54</td><td>60</td></tr><tr><td>60</td><td>56</td><td>62</td></tr></table>	Epoch	train_accuracy	val_accuracy	0	0	0	10	10	20	20	25	40	30	45	55	40	52	58	50	54	60	60	56	62
Epoch	train_accuracy	val_accuracy																							
0	0	0																							
10	10	20																							
20	25	40																							
30	45	55																							
40	52	58																							
50	54	60																							
60	56	62																							
BN-Inception	<p>bninception_1024_k1_max_train_vs_val_accuracy (top-1)</p> <table><caption>Approximate data for BN-Inception accuracy</caption><tr><th>Epoch</th><th>train_accuracy</th><th>val_accuracy</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>10</td><td>40</td><td>60</td></tr><tr><td>20</td><td>70</td><td>75</td></tr><tr><td>30</td><td>82</td><td>80</td></tr><tr><td>40</td><td>84</td><td>80</td></tr><tr><td>50</td><td>85</td><td>80</td></tr><tr><td>60</td><td>85</td><td>80</td></tr></table>	Epoch	train_accuracy	val_accuracy	0	0	0	10	40	60	20	70	75	30	82	80	40	84	80	50	85	80	60	85	80
Epoch	train_accuracy	val_accuracy																							
0	0	0																							
10	40	60																							
20	70	75																							
30	82	80																							
40	84	80																							
50	85	80																							
60	85	80

Figure 4.5. Comparison of accuracy plot for local and weighted features using base model ResNet, DenseNet and BN-Inception.

4.4.4 Temporal Dynamics

We evaluated the performance of encoding short range and medium range temporal dynamics for each segment sampled from a full video, by varying the temporal stride range from 1 to 7. All the experiments are implemented using ResNet-34 as the base model with applied weighted feature representations.

Table 4.4 presented the classification prediction for encoding different ranges of local dynamics by changing the size of temporal stride for each segment's frame sampling. Sampling from more video segments provides more reliable results as we can obtain a better representation of the global mean and for prediction averaging. Utilizing a longer temporal stride provides more information in the local dynamics, which leads to a better performance than using the stride size of 1. Overall, the results in Table 4.4 shows that consistent performance can be achieved when encoding a varying range of local dynamics using the same architecture structure. This experiment demonstrates that our model has high learning capacity and flexibility to learn short and medium temporal dynamics in a full video.

Table 4.4 Classification prediction accuracy (%) for base model ResNet-34 with varying temporal dynamics. Bold values show the best performance for the different number of segments.

Number of segments	Temporal stride 1	Temporal stride 3	Temporal stride 5	Temporal stride 7
3	83.82	84.30	84.99	85.12
5	85.06	85.75	85.28	85.25
7	85.09	85.36	85.30	85.88

4.5 Comparison with State-Of-The-Art

From the above experiments, we have evaluated: 1) the kernel size and the number of kernels for our 3D convolution layer, 2) the global mean and weighted features computation, and 3) encoding different ranges of local

dynamics with the same network. In this section, we apply the learned configurations to train our network for the three splits of UCF-101 dataset. We chose Resnet-34 as our base model and apply a 3D convolution layer with 1024 kernels with dimension $(3 \times 1 \times 1)$, followed by a global max pooling of $(1 \times 5 \times 5)$. Output spatio-temporal features are aggregated before connecting to a fully connected layer for class prediction. During training and validation, we split the video into 7 segments and sampled frames with temporal stride 7. We trained the model for 60 epochs and saved the best validated model for testing. During testing, we split the video into 7 segments, and evaluated on 10 crops (4 corners with its center, and their horizontal flip) of the sampled frames. The scores are averaged for the 10 crops and later fused for the final prediction.

The performances of our model and state-of-the-art methods are stated in Table 4.5. For fair comparison, we benchmark our model with methods that only take RGB input. There are dense sampling methods that encode local and global dynamics using stack of consecutive frames [122, 124, 134] or full video sequences [127]. Bilen, et al. [127] obtained the best performance by learning global dynamics from full video sequences for action prediction, achieving top-1 accuracy of 89.1%.

Methods that employ the sparse sampling approach (including ours), exploit a varying number of divided segments and sampled frames as network input. Diba, et al. [90] learns temporal encoding from 3D CNNs using 16 frames as input. Aggregation and encoding of features for all the segments are implemented for video-level prediction, with a performance of 86.3%. Wang, et al. [135] achieved a similar performance of 86.4% with inputs of a single frame per segment, and evaluated on videos with 25 segments. Our method sampled 3 frames from 7 segments for spatio-temporal encoding, reaching a performance of 86.5%. Our network outperforms 3D CNN architectures with dense sampling and achieves comparable results as sparse sampling, utilizing only 3 frames per segment to encode local spatio-temporal features, in a single 3D convolution layer. In contrast to the full-video encoding method that

achieved the best performance, the sparse sampling strategy is more robust to noise and has a much lower computational cost, with the trade-off in prediction accuracy.

Table 4.5 Performance comparison with state-of-the-art methods on UCF-101 dataset.

Author	Method	Input	UCF-101
Tran, et al. [122]	3D CNN	Video sequence (16 frames)	82.3
Diba, et al. [134]	Temporal transition layer in 3D CNNs	Video sequence (32 frames)	81.3
Varol, et al. [124]	Long-term temporal 3D CNNs	Video sequence (100 frames)	82.4
Bilen, et al. [127]	Dynamic Image network	Video sequence (full video sequence)	89.1
Diba, et al. [90]	3D CNNs and temporal linear encoding	Video sequence (16 frames $\times$ 3 segments)	86.3
Wang, et al. [135]	Temporal segment networks	Video sequence (1 frame $\times$ 25 segments)	86.4
Our approach	Spatio-temporal aggregation network	Video sequence (3 frames $\times$ 7 segments)	86.5

4.5.1 Qualitative Results and Visualization

This section presents the qualitative results for action prediction using local features and weighted features (experiments in Section 4.4.3), as well as sample illustrations of these features. Input frames are sampled from three segments of the video, with three consecutive frames from each segment (as shown in the top row of each figure in Figure 4.6). Prediction results displayed in Figure 4.6 are computed using base model BN-Inception with local features and weighted features. The top-5 prediction scores for both local and weighted features are normalized and presented as a bar chart in the second row of Figure 4.6 for easy reference and comparison. Features shown are obtained before and after the weighted aggregation function is applied to the weighted model, to illustrate the effect of the weighted function (third and fourth rows of Figure 4.6). The feature values are normalized and displayed in

a matrix of three rows to indicate the local/ weighted features for the three segments (each row represent features from each segment).

Figures 4.6 (a – c) show examples of weighted features outperforming local features, whereas Figures 4.6 (d, e) show examples where local features perform better. Both models have correct predictions in Figure 4.6 (f), while both models fail to predict the top-1 correct action in Figure 4.6 (g). For the features comparison, weighted features manage to suppress low-valued weights (in dark color) while maintaining the high-valued weights (in bright / light color). This illustration supports our assumption that weighted representation could provide better discriminative power than non-weighted representation, where it suppresses noise features while making important features more perceptible.

Figure 4.6. Examples of qualitative results and features visualization for model BN-Inception using local and weighted features.

4.6 Conclusion

In this chapter, we proposed a weighted single-stream spatio-temporal network which can easily adopt and extend a pretrained 2D CNN model for spatial-temporal features encoding, by replacing the 2D global pooling layer to a 3D convolution and max pooling layer. Aggregation of local features with the global mean shows improved performance for better discrimination of local motions from different action sequences. Our framework has high flexibility in encoding short and medium range local dynamics, without increasing the number of input frames and affecting the network architecture. In future work, we would look into encoding the temporal dynamics of a full video in a hierarchical representation from short, medium to long range. We could also explore different aggregation functions to learn better local and global feature representations.

CHAPTER 5

EXTENDED SPATIO-TEMPORAL NETWORK WITH TEMPORAL PYRAMID ENCODING AND GLOBAL FEATURES AGGREGATION

5.1 Introduction

This chapter extends the research in the single-stream spatio-temporal network from Chapter 4 to further improve on its learning capacity and spatio-temporal encoding. Empirical studies in the previous chapter have demonstrated that our sparse-sampling compact architecture outperforms 3D CNN with dense sampling and achieved comparable performance as spatial networks with sparse sampling. This chapter seeks to address the future work mentioned in the previous chapter, while investigating higher complexity models to boost its recognition performance.

The work in this chapter focuses on exploiting: 1) additional 3D convolution layers for varying input frames, 2) learning global representation for feature aggregation, and 3) encoding local dynamics with hierarchical temporal strides. Similarly, evaluation experiments are conducted on the UCF-101 dataset, and the results are compared with the baseline model from Chapter 4. Analyses on the model performance, training efficiency and computational cost are also discussed at the end of this chapter.

5.2 Extended Framework

The model architecture in this chapter shares the same structure as illustrated in Section 4.3.1. It adopts a pretrained 2D network for spatial feature extraction, while exploring different configurations for 3D convolution layers, global feature representation and hierarchical temporal encoding.

5.2.1 Additional 3D convolution layers

In the previous chapter, we utilized only a single 3D convolution layer for spatio-temporal encoding, as illustrated in Figure 5.1(a). This section investigates the configuration of two 3D convolution layers, applying on the architecture with different number of input frames – 3 and 5 frames, as illustrated in Figure 5.1(b, c). For both configurations, we also investigate the effects of adding a global pooling layer or directly using the convolution outputs as dense feature representations.

Figure 5.1. Structure of 3D convolution layers for: (a) a single layer with 3 input frames, (b) two layers with 3 input frames, and (c) two layers with 5 input frames.

Table 5.1 and 5.2 describe the configurations of the 3D convolution filters used for input of 3 frames and 5 frames respectively. Both tables present 5 different configurations depending on the kernel sizes used and whether a global pooling layer is added in the network. The naming of the configurations is given based on the spatial kernel sizes used, and whether it has a pooling layer or dense representation, such as '*l3 dense*', '*l5 max*' and '*l7*'.

The dimension of the input feature for Table 5.1 is given as $(c \times 3 \times 7 \times 7)$, where c denotes the channel size, 3 is the temporal depth (number of consecutive frames), and (7×7) is the spatial output. After the first temporal convolution with 1024 kernel of size $(3 \times 1 \times 1)$, the output dimension will be $(1024 \times 1 \times 7 \times 7)$. The second layer then performs 3D convolution with filter

sizes $(1 \times 3 \times 3)$, $(1 \times 5 \times 5)$ and $(1 \times 7 \times 7)$, which indicates that the second layer only performs convolution in the spatial dimension.

For Table 5.2, the input size is $(c \times 5 \times 7 \times 7)$, where 5 represents the number of input frames, and the output dimension after the first temporal convolution layer is $(1024 \times 3 \times 7 \times 7)$. The second convolution layer performs spatio-temporal convolution with kernel sizes $(3 \times 3 \times 3)$, $(3 \times 5 \times 5)$ and $(3 \times 7 \times 7)$.

Table 5.1 Configurations of different 3D convolution kernel sizes and pooling layer for input of 3 frames. The first layer is fixed for all configurations, and the second layer performs spatial convolution.

Layer	<i>l3 max</i>	<i>l3 dense</i>	<i>l5 max</i>	<i>l5 dense</i>	<i>l7</i>
3D Convolution 1	$(3 \times 1 \times 1) \times 1024$	$(3 \times 1 \times 1) \times 1024$	$(3 \times 1 \times 1) \times 1024$	$(3 \times 1 \times 1) \times 1024$	$(3 \times 1 \times 1) \times 1024$
3D Convolution 2	$(1 \times 3 \times 3) \times 1024$	$(1 \times 3 \times 3) \times 1024$	$(1 \times 5 \times 5) \times 1024$	$(1 \times 5 \times 5) \times 1024$	$(1 \times 7 \times 7) \times 1024$
3D Pooling	$(1 \times 5 \times 5)$ max pool	—	$(1 \times 3 \times 3)$ max pool	—	—
Dense layer	—	25600-D vector	—	9216-D vector	1024-D vector
Classification layer	Dropout 0.8, 101-D fully-connected layer				

Table 5.2 Configurations of different 3D convolution kernel sizes and pooling layer for input of 5 frames. The first layer is fixed for all configurations, and the second layer performs spatio-temporal convolution.

Layer	<i>l3 max</i>	<i>l3 dense</i>	<i>l5 max</i>	<i>l5 dense</i>	<i>l7</i>
3D Convolution 1	$(3 \times 1 \times 1) \times 1024$	$(3 \times 1 \times 1) \times 1024$	$(3 \times 1 \times 1) \times 1024$	$(3 \times 1 \times 1) \times 1024$	$(3 \times 1 \times 1) \times 1024$
3D Convolution 2	$(3 \times 3 \times 3) \times 1024$	$(3 \times 3 \times 3) \times 1024$	$(3 \times 5 \times 5) \times 1024$	$(3 \times 5 \times 5) \times 1024$	$(3 \times 7 \times 7) \times 1024$
3D Pooling	$(1 \times 5 \times 5)$ max pool	—	$(1 \times 3 \times 3)$ max pool	—	—
Dense layer	—	25600-D vector	—	9216-D vector	1024-D vector
Classification layer	Dropout 0.8, 101-D fully-connected layer				

5.2.2 Global Feature Representation

This section explores and compares four alternative methods in computing global representation for a full video, as well as applying a

weighted aggregation function to learn discriminative local features. The four types of global features, as shown in Figure 5.2, are categorized as:

- 1) Average pooling (mean) – the global representation is obtained by computing the mean of local spatio-temporal features, as described in Section 4.3.3.
- 2) Average pooling with FC layer – mean features obtained from approach 1) are connected to a fully-connected layer to obtain weighted mean features.
- 3) Concatenated features with FC layer – all segmental spatio-temporal features are concatenated and passed to a fully-connected layer to form the global representation.
- 4) Stacked features with 3D convolution and FC layer – local spatio-temporal features obtained from the first 3D convolution layer are stacked and convolved with a 3D kernel of size $(3 \times 3 \times 3)$, followed by global maximum pooling to obtain a 1-D representation. This output is then passed to a fully-connected layer to learn the global representation.

Figure 5.2. Illustrations of four global representations used: 1) mean features, 2) weighted mean features, 3) weighted features concatenation, and 4) weighted 3D convolution features.

The first two global representations with mean aggregated features retain our network's flexibility to train different number of input segmental frames. For concatenated and convoluted features, the input dimension has to be fixed. The network architecture has to be modified and re-trained if a

different number of segments is sampled, as it will affect the network's convolution kernel sizes, pooling layer kernels and its dense representation.

5.2.3 Temporal Pyramid Encoding

After investigating different configurations to obtain the local spatio-temporal features and a global representation, this section investigates the encoding of temporal dynamics in a pyramid representation by sampling frames from multiple temporal strides. In oppose to the previous chapter (Section 4.4.4) where we evaluate the performance of temporal dynamics by varying its temporal stride (1, 3, 5, or 7), this section aims to encode the temporal pyramid representation by pooling features from a range of temporal strides (such as 1, 3, and 5).

5.3 Experiments

Experiments in this chapter are conducted with the same implementation details as described before in Section 4.4.1. The following sub-sections provide evaluations and performance analyses on the extended features as mentioned earlier.

5.3.1 Evaluation on Two 3D Convolution Layers

The architecture search for our network with two 3D convolution layers have been presented in Table 5.1 and Table 5.2, for input frames of 3 and 5 respectively. We compare the number of trainable parameters for both settings in Table 5.3, using BN-Inception [131] as our network's base model.

Table 5.3 Number of trainable parameters for different configurations with input sizes of 3 frames and 5 frames, using base model BN-Inception.

Configuration	3 frames	5 frames
	# Params	# Params
<i>l3 max</i>	23.0M	41.8M
<i>l3 dense</i>	25.4M	44.3M
<i>l5 max</i>	39.7M	92.2M
<i>l5 dense</i>	40.6M	93.0M
<i>l7</i>	64.9M	167.7M

The table shows that increasing the input size from 3 to 5 frames doubles the computational cost for network training or higher. For a network with the same number of input size of 3 frames, utilizing a single 3D convolution layer requires the training of 13.5M parameters (from Table 4.2), while two 3D convolution layers require at least 10M additional parameters for training.

Performance comparisons for the two configurations are presented in Table 5.4. A video sequence is divided into 3 segments, with 3 or 5 consecutive frames sampled from each segment, resulting in a 4-dimensional input size of $(3 \times 9 \times 224 \times 224)$ or $(3 \times 15 \times 224 \times 224)$. The difference in the two input sizes is the second dimension which denotes the temporal depth, given by the number of segments * frames-per-segment.

Table 5.4 Validation results that shows top-1 and top-5 accuracies for configurations shown in Table 5.1 and Table 5.2.

Configuration	3 frames		5 frames	
	Top-1 acc (%)	Top-5 acc (%)	Top-1 acc (%)	Top-5 acc (%)
<i>l3 max</i>	77.90	95.19	79.43	95.51
<i>l3 dense</i>	82.98	96.17	83.51	96.30
<i>l5 max</i>	72.14	93.47	-	-
<i>l5 dense</i>	82.32	96.46	-	-
<i>l7</i>	46.84	79.33	-	-

From the empirical results, it is apparent that utilizing dense feature representations performs significantly better than adding a maximum pooling layer after the convolution process. Configuration '*l3 dense*' performs best among all the configurations for both inputs of 3 and 5 frames-per-segment. As the number of trainable parameters for the input size of 5 frames-per-segment is extremely large (more than 90M parameters), we are not able to train the models with our available resources, hence there are no results comparison for configurations '*l5 max*', '*l5 dense*' and '*l7*'.

To compare with the network with a single 3D convolution layer (in Chapter 4), the best performance obtained is 81.34% (from Table 4.2). By extending the network with an additional 3D convolution layer, it is able to boost the performance to 82.98%, a 1.64% increment. However, we should take note of the increase in the network complexity when building a deeper model, so that we could still train a network with optimized performance while considering its practical implementation.

The loss and accuracy plots for both settings are presented in Figure 5.3 and 5.4 to visualize the effectiveness of the training and validation performance over 60 epochs. From both figures, it can be seen that dense representations (*l3 dense* and *l5 dense*) can train better and converge faster than max pooling features with the steep slope and achieve high accuracy. However, max pooling features can generalize better to validation datasets as the gap between training and validation accuracy is smaller than the gap shown in the dense representation plots.

Configurations	Loss
<i>l3 max</i>	<p>Training vs validation loss</p> <p>This line graph shows the training and validation loss for the 'l3 max' configuration over 60 epochs. The y-axis represents 'Loss' from 1.0 to 4.5, and the x-axis represents 'Epoch' from 0 to 60. The training loss (blue line) starts at approximately 4.6 and decreases to about 1.0. The validation loss (orange line) starts at approximately 4.5 and decreases to about 0.9. Both losses show a significant drop in the first 10 epochs and then stabilize.</p>
<i>l3 dense</i>	<p>Training vs validation loss</p> <p>This line graph shows the training and validation loss for the 'l3 dense' configuration over 60 epochs. The y-axis represents 'Loss' from 1 to 4, and the x-axis represents 'Epoch' from 0 to 60. The training loss (blue line) starts at approximately 4.5 and decreases to about 0.5. The validation loss (orange line) starts at approximately 4.3 and decreases to about 0.7. The training loss continues to decrease more slowly than the validation loss after epoch 20.</p>
<i>l5 max</i>	<p>Training vs validation loss</p> <p>This line graph shows the training and validation loss for the 'l5 max' configuration over 60 epochs. The y-axis represents 'Loss' from 1.0 to 4.5, and the x-axis represents 'Epoch' from 0 to 60. The training loss (blue line) starts at approximately 4.6 and decreases to about 1.4. The validation loss (orange line) starts at approximately 4.5 and decreases to about 1.1. Both losses show a significant drop in the first 10 epochs and then stabilize.</p>

Configurations	Top-1 accuracy
<i>l3 max</i>	<p data-bbox="625 258 1187 281">bninception_1024_l3_max_train_vs_val_accuracy (top-1)</p> <p data-bbox="553 464 570 541">Accuracy</p> <p data-bbox="878 747 927 768">Epoch</p> <p data-bbox="716 310 797 342">train_accuracy</p> <p data-bbox="716 321 797 352">val_accuracy</p>
<i>l3 dense</i>	<p data-bbox="625 789 1187 812">bninception_1024_l3_flatten_train_vs_val_accuracy (top-1)</p> <p data-bbox="553 995 570 1073">Accuracy</p> <p data-bbox="878 1278 927 1299">Epoch</p> <p data-bbox="1040 1194 1122 1226">train_accuracy</p> <p data-bbox="1040 1215 1122 1247">val_accuracy</p>
<i>l5 max</i>	<p data-bbox="625 1320 1187 1344">bninception_1024_l5_max_train_vs_val_accuracy (top-1)</p> <p data-bbox="553 1526 570 1604">Accuracy</p> <p data-bbox="878 1810 927 1831">Epoch</p> <p data-bbox="716 1373 797 1404">train_accuracy</p> <p data-bbox="716 1383 797 1415">val_accuracy</p>

Figure 5.3. Loss and accuracy plots for different configurations network with input size of 3 frames.

Configurations	Loss
<i>l3 max</i>	<p>The graph for the <i>l3 max</i> configuration shows training loss (blue line) and validation loss (orange line) over 60 epochs. The y-axis ranges from 0.5 to 4.5. Both losses start at approximately 4.6 at epoch 0. The training loss decreases steadily to about 0.75 by epoch 60. The validation loss decreases more rapidly initially, reaching about 1.2 by epoch 20, and then fluctuates between 0.8 and 1.0 for the remainder of the training process.</p>
<i>l3 dense</i>	<p>The graph for the <i>l3 dense</i> configuration shows training loss (blue line) and validation loss (orange line) over 60 epochs. The y-axis ranges from 1 to 4.5. Both losses start at approximately 4.6 at epoch 0. The training loss decreases steadily to about 0.4 by epoch 60. The validation loss decreases to about 0.8 by epoch 10 and then remains relatively stable, fluctuating between 0.7 and 0.8 for the rest of the training process.</p>

Figure 5.4. Loss and accuracy plots for different configurations network with input size of 5 frames.

5.3.2 Evaluation on Global Representations

The experiments in this section are implemented using an input size of 3 frames per segment, with configurations '*l3 dense*' and '*l5 dense*'. We first investigate the number of trainable parameters with relation to the global representation used (as described in Section 5.2.2), as displayed in Table 5.5.

Table 5.5 Number of trainable parameters for network when different global representations were used for feature aggregation.

Global feature aggregation function	<i>l3 dense</i>	<i>l5 dense</i>
	# Params	# Params
Mean	25.4M	40.6M
Mean + FC	680.8M	125.5M
Concatenate + FC	1991.6M	295.4M
Conv3D + FC	80.0M	78.3M

The number of trainable parameters increased tremendously when we added a fully-connected layer for both mean features and concatenated features, as each neuron at the dense layer is connected to all the neurons in the fully-connected layer, which in our case is a 1024-D fully-connected layer. With dense feature output of 25600-D for ‘l3’ configuration (from Table 5.2), it has more than 5 times the trainable parameters as compared to the ‘l5’ configuration of 9216-D dense features. Validation results are presented in Table 5.6 for comparison.

Table 5.6 Validation results for different global feature aggregation functions.

Global aggregation function	<i>l3 dense</i>		<i>l5 dense</i>	
	Top-1 acc (%)	Top-5 acc (%)	Top-1 acc (%)	Top-5 acc (%)
Mean	83.93	95.96	82.87	96.19
Mean + FC	-	-	69.68	91.62
Concatenate + FC	-	-	81.21	95.77
Conv3D + FC	83.43	96.33	83.88	94.14

Aggregation functions that apply both average pooling and convolution 3D provide comparative performance of 83% in the top-1 accuracy, while top-5 accuracy achieved performance of 96%. This empirical study demonstrates that building higher complexity network does not correlates to more effective learning with better performance. A simple and effective model is more reasonable and highly favorable for practical implementation. A flexible network that can adopt different number of segmental inputs is also preferred

as we can train a single architecture and evaluate its performance with different number of input frames. Performance plots for both configurations are displayed in Figure 5.5 and Figure 5.6 for comparison. Computing global representation with mean features provides better training as shown in the steeper slope in the training and validation plots. Although the aggregation function '*Conv3D + FC*' also shows similar performances, it has a much higher number of training parameters as reported in Table 5.5. The training for the other two aggregation functions '*Mean + FC*' and '*Concatenate + FC*' did not go well due to the very large number of trainable parameters.

Global aggregation function	Loss																								
Mean	<p>Training vs validation loss</p> <p>This graph displays the training and validation loss for the Mean aggregation function. The x-axis represents the number of epochs from 0 to 60, and the y-axis represents the loss from 0 to 4. The training loss (blue line) starts at approximately 4.5 and decreases steadily to about 0.2 by epoch 60. The validation loss (orange line) starts at approximately 4.2, drops sharply to about 1.0 by epoch 10, and then fluctuates slightly around 0.8 for the remainder of the training process.</p> <table><tr><th>Epoch</th><th>train_loss</th><th>val_loss</th></tr><tr><td>0</td><td>4.5</td><td>4.2</td></tr><tr><td>10</td><td>1.0</td><td>1.0</td></tr><tr><td>20</td><td>0.5</td><td>0.8</td></tr><tr><td>30</td><td>0.3</td><td>0.8</td></tr><tr><td>40</td><td>0.2</td><td>0.8</td></tr><tr><td>50</td><td>0.2</td><td>0.8</td></tr><tr><td>60</td><td>0.2</td><td>0.8</td></tr></table>	Epoch	train_loss	val_loss	0	4.5	4.2	10	1.0	1.0	20	0.5	0.8	30	0.3	0.8	40	0.2	0.8	50	0.2	0.8	60	0.2	0.8
Epoch	train_loss	val_loss																							
0	4.5	4.2																							
10	1.0	1.0																							
20	0.5	0.8																							
30	0.3	0.8																							
40	0.2	0.8																							
50	0.2	0.8																							
60	0.2	0.8																							
Conv3D + FC	<p>Training vs validation loss</p> <p>This graph displays the training and validation loss for the Conv3D + FC aggregation function. The x-axis represents the number of epochs from 0 to 60, and the y-axis represents the loss from 0 to 4. The training loss (blue line) starts at approximately 4.5 and decreases steadily to about 0.2 by epoch 60. The validation loss (orange line) starts at approximately 4.2, drops sharply to about 1.0 by epoch 10, and then fluctuates slightly around 0.7 for the remainder of the training process.</p> <table><tr><th>Epoch</th><th>train_loss</th><th>val_loss</th></tr><tr><td>0</td><td>4.5</td><td>4.2</td></tr><tr><td>10</td><td>1.0</td><td>1.0</td></tr><tr><td>20</td><td>0.5</td><td>0.7</td></tr><tr><td>30</td><td>0.3</td><td>0.7</td></tr><tr><td>40</td><td>0.2</td><td>0.7</td></tr><tr><td>50</td><td>0.2</td><td>0.7</td></tr><tr><td>60</td><td>0.2</td><td>0.7</td></tr></table>	Epoch	train_loss	val_loss	0	4.5	4.2	10	1.0	1.0	20	0.5	0.7	30	0.3	0.7	40	0.2	0.7	50	0.2	0.7	60	0.2	0.7
Epoch	train_loss	val_loss																							
0	4.5	4.2																							
10	1.0	1.0																							
20	0.5	0.7																							
30	0.3	0.7																							
40	0.2	0.7																							
50	0.2	0.7																							
60	0.2	0.7

Figure 5.5. Loss and accuracy plots for global aggregation functions applied on *l3 dense* network.

Global aggregation function	Loss
Mean	<p>The graph shows training loss (blue line) and validation loss (orange line) over 60 epochs. Training loss starts at approximately 4.5 and decreases steadily to about 0.5. Validation loss starts at approximately 4.5, drops sharply to about 1.5 by epoch 10, and then fluctuates between 0.8 and 1.2 for the remainder of the training process.</p>
Mean + FC	<p>The graph shows training loss (blue line) and validation loss (orange line) over 60 epochs. Both losses start at approximately 4.5. Training loss decreases to about 1.5 by epoch 30 and then continues to decrease slowly to about 1.3. Validation loss decreases more rapidly than training loss, reaching about 1.5 by epoch 30 and then fluctuating between 1.1 and 1.4 for the remainder of the training process.</p>

Global aggregation function	Top-1 accuracy																								
Mean	<p>bninception_1024_l5f_weighted_mean_train_vs_val_accuracy (top-1)</p> <table><caption>Approximate data for Mean aggregation function</caption><tr><th>Epoch</th><th>train_accuracy</th><th>val_accuracy</th></tr><tr><td>0</td><td>5</td><td>10</td></tr><tr><td>10</td><td>55</td><td>65</td></tr><tr><td>20</td><td>85</td><td>80</td></tr><tr><td>30</td><td>90</td><td>82</td></tr><tr><td>40</td><td>90</td><td>82</td></tr><tr><td>50</td><td>90</td><td>82</td></tr><tr><td>60</td><td>90</td><td>82</td></tr></table>	Epoch	train_accuracy	val_accuracy	0	5	10	10	55	65	20	85	80	30	90	82	40	90	82	50	90	82	60	90	82
Epoch	train_accuracy	val_accuracy																							
0	5	10																							
10	55	65																							
20	85	80																							
30	90	82																							
40	90	82																							
50	90	82																							
60	90	82																							
Mean + FC	<p>bninception_1024_l5f_weighted_meanfc_train_vs_val_accuracy (top-1)</p> <table><caption>Approximate data for Mean + FC aggregation function</caption><tr><th>Epoch</th><th>train_accuracy</th><th>val_accuracy</th></tr><tr><td>0</td><td>2</td><td>5</td></tr><tr><td>10</td><td>5</td><td>10</td></tr><tr><td>20</td><td>25</td><td>35</td></tr><tr><td>30</td><td>55</td><td>60</td></tr><tr><td>40</td><td>60</td><td>65</td></tr><tr><td>50</td><td>63</td><td>68</td></tr><tr><td>60</td><td>65</td><td>70</td></tr></table>	Epoch	train_accuracy	val_accuracy	0	2	5	10	5	10	20	25	35	30	55	60	40	60	65	50	63	68	60	65	70
Epoch	train_accuracy	val_accuracy																							
0	2	5																							
10	5	10																							
20	25	35																							
30	55	60																							
40	60	65																							
50	63	68																							
60	65	70

Figure 5.6. Loss and accuracy plots for global aggregation functions applied on *l5* dense network.

5.3.3 Evaluation on Temporal Pyramid Encoding

The experiments in this section are implemented on the network in Chapter 4 with a weighted single 3D convolution layer. We utilize a partial batch-norm training strategy that freezes the parameters in all the batch-norm layers except for the top layer in each local temporal stride. We conducted two experiments for the temporal pyramid with stride (1, 3, 5) and (3, 5, 7), using base model BN-Inception and Resnet-34. Table 5.7 shows the number of

trainable parameters for both models, while validation results are presented in Table 5.8 and Table 5.9. The results in the two tables demonstrate that utilizing the max pooling layer to encode local pyramid features works better than averaging the features from the local temporal strides. This suggests that important features extracted at different ranges of temporal dynamics are crucial to learning discriminative motion for improved prediction. Another advantage of using pyramid temporal strides is that it does not affect the number of trainable parameters in the architecture (it has the same number of trainable parameters with a single-stride architecture as shown in Chapter 4, Table 4.3).

Table 5.7 Number of trainable parameters for temporal pyramid encoding using base model BN-Inception and Resnet-34 for a single 3D convolution layer network.

Pyramid temporal strides	BN-Inception	ResNet-34
	# Params	# Params
1, 3 and 5	13.5M	23.0M
3, 5 and 7	13.5M	23.0M

Table 5.8 Validation results for temporal pyramid encoding using base model BN-Inception.

Local pooling	Temporal stride (1, 3 and 5)		Temporal stride (3, 5 and 7)	
	Top-1 acc (%)	Top-5 acc (%)	Top-1 acc (%)	Top-5 acc (%)
Average pooling	83.95	96.43	83.40	96.19
Max pooling	84.19	97.15	84.01	96.59

Table 5.9 Validation results for temporal pyramid encoding using base model ResNet-34.

Local pooling	Temporal stride (1, 3 and 5)		Temporal stride (3, 5 and 7)	
	Top-1 acc (%)	Top-5 acc (%)	Top-1 acc (%)	Top-5 acc (%)
Average pooling	84.35	96.51	84.38	96.19
Max pooling	84.91	96.67	85.17	96.54

5.4 Comparison with Baseline Model and Analysis

After conducting experiments on the extended model with 1) additional 3D convolution layers, 2) global features aggregation and 3) temporal pyramid encoding, this section compares the performance of the

extended features and the baseline model (from Chapter 4) on the training efficiency and classification accuracy.

5.4.1 Adding 3D convolution layers and number of input frames

Table 5.10 presents the performance obtained from the best configuration from each model for comparison. We study the effect of adding more 3D convolution layers and using different number of input frames for model training. The results in Table 5.10 shows that adding more 3D convolution layers and increasing the number of input frames per segment leads to a higher prediction accuracy. However it requires more computational resources as the number of trainable parameters increases. The accuracy is boosted from 81.34% to 83.51%, a 2.17% increment for an additional of 30.8M trainable parameters. Depending on the application and requirement of the network, tradeoffs between practical computational cost and good prediction accuracy have to be considered. The network configurations and training hyper-parameters conducted in our experiments are not fully-optimized, hence the performance can still be further enhanced.

Table 5.10 Performance comparison for network using one or more 3D convolution layers and input of 3 or more frames.

# Conv3D layers	# Input frames per segment	Configuration	# Params	Top-1 acc (%)	Top-5 acc (%)
1	3	<i>k1</i>	13.5M	81.34	95.96
2	3	<i>l3 dense</i>	25.4M	82.98	96.17
2	5	<i>l3 dense</i>	44.3M	83.51	96.30

5.4.2 Global features aggregation

The previous empirical study in Section 4.4.3 demonstrated that applying a weighted aggregation function on the local features outperform networks that directly perform predictions on local features. We proposed a simple global representation obtained by averaging all the segmental features, and perform element-wise multiplication on the local features to compute the weighted representations.

This chapter investigates other possible global representations that can be learned from the training process, such as adding a convolution layer, having a fully-connected layer, or concatenating all the local features. From the evaluations in Section 5.3.2, it is apparent that adding a fully-connected layer to learn a global representation imposes a high number of training parameters (up to 2000M) which is beyond our computational capability. A high complexity model may face under-fitting problems which lead to a worse performance than the baseline model.

Table 5.11 shows the performance of a network with two 3D convolution layers, using local features only and using mean global features as weighted representation. Utilizing a mean global representation from average pooling local features (as described in Section 4.3.3) works best among other proposed methods in Section 5.2.2, since it does not impose additional training parameters to the network and yet is able to boost about 1% of the prediction accuracy.

Table 5.11 Performance comparison for network using local features only or with global features aggregation.

# Conv3D layers	# Input frames per segment	Configuration	# Params	Global aggregation function	Top-1 acc (%)
2	3	<i>l3 dense</i>	25.4M	-	82.98
2	3	<i>l3 dense</i>	25.4M	Mean	83.93
2	3	<i>l5 dense</i>	40.6M	-	82.32
2	3	<i>l5 dense</i>	40.6M	Mean	82.87

5.4.3 Temporal pyramid encoding

This sub-section discusses the encoding of local temporal dynamics using varying temporal strides. To compare the results of temporal pyramid encoding with Section 4.4.4, Table 5.12 presents the results implemented on a network with a single 3D convolution layer, using base model Resnet-34. The full video is divided into 3 segments before frames sampling. The number of trainable parameters is the same for all the networks, regardless of the size of temporal strides.

The networks in Table 5.12 show comparable performance using either a fixed temporal stride or a hierarchy of temporal strides. Hierarchical encoding of 3 different temporal strides performs slightly better than the average performance of a single stride network. For example, the average of performance for fixed temporal strides 3, 5 and 7 provides accuracy of 84.80%, while the pyramid encoding of these 3 strides provides a slightly better performance of 85.17%. The difference is that a pyramid encoding network computes the maximum pooling on the local features obtained from the 3 different strides before computing its weighted features for final prediction.

Table 5.12 Performance comparison for network with varying temporal strides applying on base model ResNet-34.

Temporal stride	Top-1 acc (%)	Top-5 acc (%)
1	83.82	96.56
3	84.30	96.64
5	84.99	96.83
7	85.12	96.83
1, 3 and 5	84.91	96.67
3, 5 and 7	85.17	96.54

5.5 Features Visualization and Qualitative Results

In Section 5.3.1, experiments were conducted using different configurations of the BN-Inception model. Here we illustrate the outcomes using the results of the *l5 dense* configuration. Example illustrations of spatial

features and spatio-temporal features from the model and the qualitative results are presented in Figure 5.7, for six different actions. Three consecutive frames from three segments of the video are sampled as the network input (top row in each of the figures of Figure 5.7). Spatial features are obtained from the pretrained 2D model with 1024 outputs of dimension of $(9 \times 7 \times 7)$. We randomly pick one of these outputs and display its (7×7) spatial feature in the second row in Figure 5.7, one for each of the nine frames. After the first 3D convolution layer, we obtain 1024 spatio-temporal outputs with dimension $(3 \times 7 \times 7)$, three sets of which are shown in the third row in Figure 5.7. The second 3D convolution layer produces 1024 outputs of size $(3 \times 3 \times 3)$, three sets of which are shown in the fourth row of Figure 5.7.

Figures 5.7 (a – d) show examples of the qualitative results and features visualization for the model with the correct prediction, while Figures 5.7 (e, f) illustrate the prediction results for the model with incorrect top-1 prediction. From the feature visualization, spatio-temporal features show a more complicated representation where temporal information is encoded. Although it is difficult to interpret the representations visually, we can still observe that the patterns look consistent across the segments.

Figure 5.7. Examples of qualitative results and features visualization for BN-Inception model with *l5 dense* configuration.

5.6 Conclusion

This chapter explores a number of extended features that can be applied on the baseline model of the spatio-temporal network introduced in Chapter 4. We study the extended framework in three different aspects: 1) the number of 3D convolution layers, 2) global features aggregation function and 3) encoding hierarchical temporal strides. Besides comparing the performance in prediction accuracy, we also consider the practical implementation with reasonable computational cost and effective training. Building a complex network with deeper layers or higher trainable parameters does not guarantee better performance as it may lead to under- or over-fitting problems. Tuning of hyper-parameters to optimize a network is also a non-trivial task as it requires multiple trial-and-error experiments or applying different training strategies. The experiments conducted in Chapters 4 and 5 share the same training hyper-parameters for comparison reference, but are not fully-optimized. As our baseline model is compact with effective learning capacity, there are still rooms for further extension and improvement to enhance its prediction accuracy.

After obtaining promising results from our spatio-temporal network that adopts pre-trained 2D CNN for spatial features extraction, followed by 3D convolution layers for spatio-temporal encoding, we propose to generalize this idea to pre-trained CNN models while preserving its network's depth and properties. Instead of adding 3D convolution layers, we modify its existing 2D convolution layers to perform temporal encoding (with 1-D kernels) and spatio-temporal encoding (with 3-D kernels). The details of the framework will be explained in the next chapter.

CHAPTER 6

SEMI-CNN ARCHITECTURE FOR EFFECTIVE SPATIO-TEMPORAL LEARNING IN ACTION RECOGNITION

6.1 Introduction

After investigating the effectiveness of a single stream spatio-temporal network in action recognition, this chapter aims to generalize the concept in existing CNN models while retaining their network properties to further improve on the learning capacity and prediction performance. We introduce a fusion approach that combines the spatial layers in 2D CNN and spatiotemporal layers in 3D CNN, namely “semi-CNN”. Transfer learning is adopted to extract mid-level spatial features, followed by applying temporal encoding and spatio-temporal convolutions on the stack of representations. We validate the performance of semi-CNN by adopting its fusion method onto three popular models: VGG [136], ResNet [130] and DenseNet [133]. We utilize pre-trained models on ImageNet to initialize our 2D convolutional layers and perform fine-tuning on the additional 1D and 3D convolution layers. Our empirical results evaluated on the action recognition dataset UCF-101 demonstrate that fusing 1D, 2D and 3D convolutions outperforms its 3D model of the same depth.

6.2 Related Work

Different fusion methods have been investigated, such as fusing two-stream CNN and 3D CNN [92], mixing 2D and 3D convolutions in an architecture [137], and splitting 3D convolution to 2D spatial and 1D temporal convolutions [138]. These methods have demonstrated improved

performances than individual architectures that utilize full 2D or 3D convolutions.

6.2.1 Full 3D CNN

Ji, et al. [77] first proposed a 3D CNN model for action recognition. Tran, et al. [122] experimented with different network architectures and kernel sizes for 3D convolutions, then presented an architecture that learns spatial features in the lower layer, and saliency motion in the higher layers. Ng, et al. [81] trained a multi-task 3D CNN model that splits into two streams to learn the motion features while recognizing actions. Hara, et al. [139] built 3D ResNets architecture, by exploiting the effectiveness of residual learning in deep 2D ResNets [130]. Varol, et al. [124] trained 3D CNNs on both image frames and motion features using multiple spatio-temporal scales, to obtain combined results.

6.2.2 Spatio-temporal fusion

The concept of factorizing 3D spatio-temporal convolution kernels to 2D and 1D kernels was introduced by Sun, et al. [80] in their Factorized Spatio-temporal Convolutional Networks. The architecture starts from 2D convolution layers for spatial features learning and split into two streams for spatial and temporal encoding. In the temporal stream, the spatial representations are transformed and permuted before passing to the temporal 1D convolution layer. Feichtenhofer and Zisserman [92] introduced a fusion of two-stream CNN and 3D CNN, where various fusions were investigated for both spatial and temporal streams.

Tran, et al. [137] introduced two variants of spatio-temporal learning. The first, named Mixed Convolutions (Figure 6.1(b)), contains both 2D and 3D convolutions in a ResNet model, where the structure starts from 3D convolution layers before connecting to 2D convolution layers (a reverse Mixed Convolutions (Figure 6.1(c)) has the structure of 2D to 3D convolutions). Mixed convolution networks shows comparable results as a 3D

ResNet model of the same depth. Another variant called “ $R(2+1)D$ ” (Figure 6.1(e)), decomposes a 3D convolution into a 2D spatial convolution and a 1D temporal convolution. By replacing all the 3D convolutional blocks to $(2+1)D$ blocks in a ResNet architecture, the performance is superior than the Mixed Convolutions and 3D ResNets.

Figure 6.1. Variants of architecture for ResNet model as described in [137]. (a) 2D ResNet, (b) ResNet with mixed convolutions, (c) reversed mixed convolutions, (d) 3D ResNet, (e) ResNet with $(2+1)D$ convolutions.

The variant of the architecture which is the most related to our proposed framework is the reversed mixed convolutions network (rMCx) [137] as shown in Figure 6.1(c). The empirical results from the study show that rMCx outperforms R3D with 0.6 - 0.9% when trained on a larger dataset, the Kinetics dataset [140], with input size of $16 \times 112 \times 112$. The difference between our proposed architecture and rMCx is our architecture performs spatial convolutions at the lower layers, followed by temporal convolutions and spatio-temporal convolutions at the higher layers. In addition, our model outperforms 3D ResNet by up to 18% in the top-1 accuracy for an input of 16 video clips when evaluated on the UCF-101 dataset. The details of our architecture are explained in the following section.

6.3 Proposed Architecture

Our proposed framework starts from selecting an existing CNN model as a base model to initialize the network architecture. Pre-trained weights for

2D convolution layers are transferred to our semi-CNN to form the bottom layers of our architecture (Figure 6.2(c)). Temporal convolution layers are then added to form the intermediate structure. Temporal encoding downsamples the depth features while preserving its spatial dimension. Lastly, spatio-temporal encoding layers are added on top before connecting to a fully-connected layer for classification prediction. Figure 6.2 illustrates the comparison of architecture designs for our semi-CNN model and 2D and 3D CNNs.

Figure 6.2. Architecture comparison for (a) 2D CNN, (b) 3D CNN and (c) our proposed semi-CNN.

6.3.1 Base Model

As much as possible, we preserve the configurations of the layers in the baseline models and retain the model's depth, while aggregating the layers to perform convolution in the spatial, temporal and spatio-temporal dimensions. The architecture and network properties of our selected base model are briefly explained in the following sub-sections.

6.3.1.1 Very Deep Convolutional Networks (VGG)

The architecture of very deep CNN, named VGG, with depth of 16 – 19 layers was introduced by Simonyan and Zisserman [136] for the task of large-scale image classification. The convolution filter size is fixed at 3×3 for all

layers to allow deeper implementation while increasing non-linearity functions (by adding a rectification layer after each convolution) to learn complex representations. After the convolution layers, the output is max pooled before connecting to three fully-connected layers of 4096-D. This leads to a high number of learnable parameters, with 138M for VGG-16 layers and 144M for VGG-19 layers.

6.3.1.2 Residual Networks (ResNets)

When developing deeper networks by stacking more and more convolution layers, a critical problem arises during back-propagation where the gradients decrease and become too small or vanish while propagating to the lower layers. This prevents weights learning and updating in the layers, which prevents the network from converging. Hence, adding more layers without any remedy will result in stagnant accuracy or even poorer performance [130]. He, et al. [130] proposed a residual learning framework, which incorporates a residual learning formulation in the stacked layers to allow better optimization. Residual learning (Figure 6.3) is implemented using “shortcut connections” where the output of a previous layer can be mapped and added to the output of the connecting stacked layers. Another benefit of residual learning is that it does not incur additional complexity or parameters to the network, which makes the development of extremely deep network (up to 152 layers) possible, with complexity much lower than VGG nets.

Figure 6.3. Residual learning block in ResNet [130].

6.3.1.3 Dense Convolutional Networks (DenseNets)

Another work proposed to build deeper architecture by connecting each convolution layer to all the subsequent layers (Figure 6.4), forming a densely connected network, DenseNet [133]. Feature maps from previous connected layers were re-used in subsequent layers, hence reducing the number of trainable parameters. Besides, the vanishing gradient problem is also addressed by the skip connections (similar to the shortcut connections in ResNets) between layers, which map the output of a previous layer to the next connecting layer, and allow the gradients from every layer to be accessed during back-propagation. DenseNet has a considerably lower number of parameters when compared with ResNet of the same depth, and yet retains a high training capacity when the model goes deeper.

Figure 6.4. A dense block with 5 connected layers. Features from lower layers are used as inputs for all sub-sequent layers [133].

6.3.2 Configuration Comparison with 2D and 3D CNN

This section presents a comparison of the architecture design and configuration for network layers in VGG, ResNet and DenseNet. Our semi-CNN performs spatial convolution first, followed by temporal convolution and finally spatio-temporal convolution. When computing the spatial convolution, the output shape of the temporal depth is preserved. Similarly, when computing the temporal convolution, the spatial dimension of the output is retained. During spatio-temporal convolution, features are convolved in all three dimensions.

With the input dimension of $16 \times 224 \times 224$, where 16 denotes the number of consecutive frames, 224×224 denotes the spatial dimension, the input will be passed through several spatial convolution blocks and pooling layers to obtain spatial features output of size $16 \times 14 \times 14$. The features are then convolved with a temporal block and the features downsampled to $8 \times 14 \times 14$, before being processed by the spatio-temporal blocks at the top layers. With spatio-temporal convolutions, the feature size is reduced to $2 \times 3 \times 3$ or $2 \times 4 \times 4$. Later, a global pooling layer will further downsample the features to a 1-D vector representation. Lastly, this representation will be connected to fully-connected layers for classification prediction.

To retain the same network depth as 2D convolution networks, we reduce the number of layers in the spatial convolution blocks and add layers to the temporal and spatio-temporal blocks. Below, Tables 6.1 to 6.3 present working examples of aggregated layers and configurations for our spatial, temporal and spatio-temporal blocks for model VGG, ResNet and DenseNet. The configurations described are not optimized, but have shown reasonably good performance when validated in the experimental section.

Table 6.1 describes the configuration for VGG-16 [136], a 3D convolution network (C3D) [122], and our semi-VGG-16. C3D has 11 layers, while VGG-16 has 16 layers in their network. Table 6.2 shows examples of the configuration for 2D ResNet [130], 3D ResNnet [139] and our semi-ResNet model with 18-layers depth. Residual learning is implemented in all the convolution blocks. Comparison of DenseNet configurations is presented in Table 6.3 for its 2D model [133], 3D model (derived from 2D network) and our semi-DenseNet model. Dense connections between layers in each block are preserved.

Table 6.1 Configuration comparison for VGG-16, C3D and our model semi-VGG-16.

Layer name	VGG-16 [136]		C3D [122]		Semi-VGG-16 (Ours)	
	Configuration	Output shape	Configuration	Output shape	Configuration	Output shape
Conv1	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$	(224, 224)	$[3 \times 3 \times 3, 64]$	(16, 224, 224)	$\begin{bmatrix} 1 \times 3 \times 3, 64 \\ 1 \times 3 \times 3, 64 \end{bmatrix}$	(16, 224, 224)
Max pool	$\begin{bmatrix} 2 \times 2 \\ \text{stride 2} \end{bmatrix}$	(112, 112)	$\begin{bmatrix} 1 \times 2 \times 2 \\ \text{stride (1, 2, 2)} \end{bmatrix}$	(16, 112, 112)	$\begin{bmatrix} 1 \times 2 \times 2 \\ \text{stride (1, 2, 2)} \end{bmatrix}$	(16, 112, 112)
Conv2	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$		$[3 \times 3 \times 3, 128]$		$\begin{bmatrix} 1 \times 3 \times 3, 128 \\ 1 \times 3 \times 3, 128 \end{bmatrix}$	
Max pool	$\begin{bmatrix} 2 \times 2 \\ \text{stride 2} \end{bmatrix}$	(56, 56)	$\begin{bmatrix} 2 \times 2 \times 2 \\ \text{stride 2} \end{bmatrix}$	(8, 56, 56)	$\begin{bmatrix} 1 \times 2 \times 2 \\ \text{stride (1, 2, 2)} \end{bmatrix}$	(16, 56, 56)
Conv3_x	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$		$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix}$		$\begin{bmatrix} 1 \times 3 \times 3, 256 \\ 1 \times 3 \times 3, 256 \\ 1 \times 3 \times 3, 256 \end{bmatrix}$	
Max pool	$\begin{bmatrix} 2 \times 2 \\ \text{stride 2} \end{bmatrix}$	(28, 28)	$\begin{bmatrix} 2 \times 2 \times 2 \\ \text{stride 2} \end{bmatrix}$	(4, 28, 28)	$\begin{bmatrix} 1 \times 2 \times 2 \\ \text{stride (1, 2, 2)} \end{bmatrix}$	(16, 28, 28)
Conv4_x	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$		$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix}$		$\begin{bmatrix} 1 \times 3 \times 3, 512 \\ 1 \times 3 \times 3, 512 \end{bmatrix}$	
Max pool	$\begin{bmatrix} 2 \times 2 \\ \text{stride 2} \end{bmatrix}$	(14, 14)	$\begin{bmatrix} 2 \times 2 \times 2 \\ \text{stride 2} \end{bmatrix}$	(2, 14, 14)	$\begin{bmatrix} 1 \times 2 \times 2 \\ \text{stride (1, 2, 2)} \end{bmatrix}$	(16, 14, 14)
Temporal conv	-				$[3 \times 1 \times 1, 512]$	
Temporal max pool	-				$\begin{bmatrix} 3 \times 1 \times 1 \\ \text{stride (2, 1, 1)} \end{bmatrix}$	(8, 14, 14)
Spatio-temporal conv	-				$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix}$	
Spatio-temporal max pool	-				$\begin{bmatrix} 2 \times 2 \times 2 \\ \text{stride 2} \end{bmatrix}$	(4, 7, 7)
Conv5_x	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$	(14, 14)	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix}$	(2, 14, 14)	$[3 \times 3 \times 3, 512]$	
Max pool	$\begin{bmatrix} 2 \times 2 \\ \text{stride 2} \end{bmatrix}$	(7, 7)	$\begin{bmatrix} 2 \times 2 \times 2 \\ \text{stride 2} \end{bmatrix}$	(1, 8, 8)	$\begin{bmatrix} 2 \times 2 \times 2 \\ \text{stride 2} \end{bmatrix}$	(2, 3, 3)
Fc6	4096-D					
Fc7	4096-D					
Fc8	101-d, softmax					

Table 6.2 Configuration comparison for 2D ResNet, 3D ResNet and our semi-ResNet, for an 18-layers network.

Layer name	2D ResNet [130]		3D ResNet [139]		Semi-ResNet (Ours)	
	18-layer	Output shape	18-layer	Output shape	18-layer	Output shape
Conv1	$[7 \times 7, 64]$ stride 2	(112, 112)	$[7 \times 7 \times 7, 64]$ stride (1, 2, 2)	(16, 112, 112)	$[1 \times 7 \times 7, 64]$ stride (1, 2, 2)	(16, 112, 112)
Max pool	$[3 \times 3]$ stride 2	(56, 56)	$[3 \times 3 \times 3]$ stride 2	(8, 56, 56)	$[1 \times 3 \times 3]$ stride (1, 2, 2)	(16, 56, 56)
Conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$		$\begin{bmatrix} 3 \times 3 \times 3, 64 \\ 3 \times 3 \times 3, 64 \end{bmatrix} \times 2$		$\begin{bmatrix} 1 \times 3 \times 3, 64 \\ 1 \times 3 \times 3, 64 \end{bmatrix} \times 2$	
Conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	(28, 28)	$\begin{bmatrix} 3 \times 3 \times 3, 128 \\ 3 \times 3 \times 3, 128 \end{bmatrix} \times 2$	(4, 28, 28)	$\begin{bmatrix} 3 \times 3 \times 3, 128 \\ 3 \times 3 \times 3, 128 \end{bmatrix}$	(16, 28, 28)
Max pool	-				$[1 \times 3 \times 3]$ stride (1, 2, 2)	(16, 14, 14)
Temporal conv block	-				$\begin{bmatrix} 3 \times 1 \times 1, 128 \\ 3 \times 1 \times 1, 128 \end{bmatrix}$	
Temporal max pool	-				$[3 \times 1 \times 1]$ stride (2, 1, 1)	(8, 14, 14)
Spatio-temporal conv block	-				$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix}$ stride 1	
Conv4_x	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	(14, 14)	$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix} \times 2$	(2, 14, 14)	$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix}$	(4, 7, 7)
Conv5_x	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	(7, 7)	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix} \times 2$	(1, 7, 7)	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix} \times 2$	(2, 4, 4)
Avg pool	$[7 \times 7]$	(1, 1)	$[1 \times 7 \times 7]$	(1, 1, 1)	$[2 \times 4 \times 4]$	(1, 1, 1)
Fc	101-d, softmax					

Table 6.3 Configuration comparison for 2D DenseNet, 3D DenseNet and our semi-DenseNet, with network depth of 121 layers.

Layer name	2D DenseNet [133]		3D DenseNet		Semi-DenseNet (Ours)	
	DenseNet-121	Output shape	DenseNet-121	Output shape	DenseNet-121	Output shape
Convolution	$[7 \times 7, 64]$ stride 2	(112, 112)	$[7 \times 7 \times 7, 64]$ stride (1,2,2)	(16, 112, 112)	$[1 \times 7 \times 7, 64]$ stride (1, 2, 2)	(16, 112, 112)
Max pool	$[3 \times 3]$ stride 2	(56, 56)	$[3 \times 3 \times 3]$ stride 2	(8, 56, 56)	$[1 \times 3 \times 3]$ stride (1, 2, 2)	(16, 56, 56)
Dense block 1	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 32 \end{bmatrix} \times 6$		$\begin{bmatrix} 1 \times 1 \times 1, 128 \\ 3 \times 3 \times 3, 32 \end{bmatrix} \times 6$		$\begin{bmatrix} 1 \times 1 \times 1, 128 \\ 1 \times 3 \times 3, 32 \end{bmatrix} \times 6$	
Transition layer 1	$[1 \times 1, 128]$ conv		$[1 \times 1 \times 1, 128]$ conv		$[1 \times 1 \times 1, 128]$ conv	
	$[2 \times 2]$ avg pool stride 2	(28, 28)	$[2 \times 2 \times 2]$ avg pool stride 2	(4, 28, 28)	$[1 \times 2 \times 2]$ avg pool stride (1, 2, 2)	(16, 28, 28)
Dense block 2	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 32 \end{bmatrix} \times 12$		$\begin{bmatrix} 1 \times 1 \times 1, 128 \\ 3 \times 3 \times 3, 32 \end{bmatrix} \times 12$		$\begin{bmatrix} 1 \times 1 \times 1, 128 \\ 1 \times 3 \times 3, 32 \end{bmatrix} \times 6$	
Transition layer 2	$[1 \times 1, 256]$ conv		$[1 \times 1 \times 1, 256]$ conv		$[1 \times 1 \times 1, 320]$ conv	
	$[2 \times 2]$ avg pool stride 2	(14, 14)	$[2 \times 2 \times 2]$ avg pool stride 2	(2, 14, 14)	$[1 \times 2 \times 2]$ avg pool stride (1, 2, 2)	(16, 14, 14)
Temporal dense block	-				$\begin{bmatrix} 1 \times 1 \times 1, 128 \\ 3 \times 1 \times 1, 32 \end{bmatrix} \times 6$	
Temporal transition layer	-				$[1 \times 1 \times 1, 256]$ conv	
					$[3 \times 1 \times 1]$ avg pool stride (2, 1, 1)	
Spatio-temporal dense block	-				$\begin{bmatrix} 1 \times 1 \times 1, 128 \\ 3 \times 3 \times 3, 32 \end{bmatrix} \times 12$	
Spatio-temporal transition layer	-				$[1 \times 1 \times 1, 640]$ conv	(4, 7, 7)
					$[2 \times 2 \times 2]$ avg pool stride 2	
Dense block 3	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 32 \end{bmatrix} \times 24$	(14, 14)	$\begin{bmatrix} 1 \times 1 \times 1, 128 \\ 3 \times 3 \times 3, 32 \end{bmatrix} \times 24$	(2, 14, 14)	$\begin{bmatrix} 1 \times 1 \times 1, 128 \\ 3 \times 3 \times 3, 32 \end{bmatrix} \times 12$	(4, 7, 7)
Transition layer 3	$[1 \times 1, 512]$ conv		$[1 \times 1 \times 1, 512]$ conv		$[1 \times 1 \times 1, 512]$ conv	
	Dense block 4	$[2 \times 2]$ avg pool stride 2	(7, 7)	$[2 \times 2 \times 2]$ avg pool stride 2	(1, 7, 7)	$[2 \times 2 \times 2]$ avg pool stride 2
$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 32 \end{bmatrix} \times 16$		$\begin{bmatrix} 1 \times 1 \times 1, 128 \\ 3 \times 3 \times 3, 32 \end{bmatrix} \times 16$		$\begin{bmatrix} 1 \times 1 \times 1, 128 \\ 3 \times 3 \times 3, 32 \end{bmatrix} \times 16$		
Global avg pool	$[7 \times 7]$	(1, 1)	$[1 \times 7 \times 7]$	(1, 1, 1)	$[2 \times 3 \times 3]$	(1, 1, 1)
Fc	101-d, softmax					

6.4 Experiments

6.4.1 Implementation details

We evaluated our framework using the popular action recognition dataset, UCF-101. It contains a total of 13320 video clips, with 101 action classes, and is divided into 3 training-validation splits. In our experiments, we only utilize the split-1 training and validation sets for evaluation. During training, 16 consecutive frames are randomly sampled from each video. Input frames are re-scaled and randomly cropped at multiple scales, before resizing them to the size of 224×224 . They are also randomly flipped horizontally to allow data augmentation for better training. The input is normalized using the mean and standard deviation of the ImageNet dataset. For validation, the input frames from each validation video are sampled at fixed locations. The frames are rescaled, and center cropped, without horizontal flipping.

Our network is trained end-to-end using stochastic gradient descent with learning rate 0.1, momentum 0.9, dampening 0.9, and weight decay of 0.0001. The learning rate is reduced by a factor of 10 when the validation loss value does not decrease for 10 epochs. The network architecture and training is implemented in PyTorch [132] with CUDA, utilizing two GPUs of GeForce RTX2080Ti. Due to memory constraints, we implemented accumulated gradients during training to allow processing of large batch sizes, while retaining the size of the computation graph. We use a mini batch size of 8 (or 4 or 2 depending on the network complexity) and accumulated the gradients for 32 iterations (effective batch size is 256) before back-propagation. The architecture is trained for 50 epochs and we presented the validation results for comparison.

6.4.2 Trainable Parameters an Configurations

This section presents the number of trainable parameters and configuration settings for each model VGG, ResNet and DenseNet (Tables 6.4 to 6.6), and compares with the architectures of full 2D, 3D and our semi-CNN

model. As our model utilizes transfer learning to initialize our network parameters, we also present the number of pre-trained parameters for each network.

For Table 6.4, we compare with 2D VGG-16 [136], a 3D convolution network (C3D) [122], and our semi-VGG-16. C3D has 11 layers, while VGG-16 has 16 layers in their networks. For ResNet in Table 6.5, we compare with 2D ResNets [130], 3D ResNets [139] and our semi-ResNets. Comparison of DenseNet is the 2D model [133], 3D model (derived from 2D network) and our semi-DenseNet model is presented in Table 6.6. The 2D convolution network takes an input of size (48, 224, 224), where its input channel size is 48, while the input dimension for 3D and our semi-convolution is (3, 16, 224, 224), with channel size 3 for each frame's RGB.

The notation for the configuration settings in Tables 6.4 to 6.6 is set as (ss, tt, st, fc) , where ss denotes the number of spatial layers with convolutional kernel $(1 \times x \times x)$, tt denotes the number of temporal encoding layers with kernel size $(x \times 1 \times 1)$, and st denotes the number of spatio-temporal layers with convolutional kernel size of $(x \times x \times x)$. The number of fully-connected layers is denoted as fc . The value of convolutional kernel size, x , is set with reference to the configurations in the 2D base model. For instance, the spatial kernel size for semi-VGG-16 is $(1 \times 3 \times 3)$, while for semi-ResNet-18, the kernel size for the first spatial convolutional block is $(1 \times 7 \times 7)$. For DenseNet model, the number of transition layers is not included in the configuration size.

From Tables 6.4 to 6.6, we can see that 3D convolution networks have a much higher number of trainable parameters as compared to their corresponding 2D convolution networks. Our semi-CNN architecture has the lowest number of parameters for VGG-16 model. As for ResNet and DenseNet models, our network has higher parameters than 2D convolution networks, but slightly lower than 3D convolution networks. The trainable parameters are obtained from the lower layers of the pre-trained network for our spatial convolutions only.

Table 6.4 Number of trainable parameters and configuration setting for VGG-16 models.

VGG-16	C3D	Semi-VGG-16		
# Params	# Params	Configuration	# Pretrained params	# Total params
134.7M	179.1M	(9, 1, 3, 3)	5.3M	82.2M

Table 6.5 Number of trainable parameters and configuration settings for ResNet models with different number of layers.

ResNet	2D-CNN	3D-CNN	Semi-CNN		
	# Params	# Params	Configuration	# Pretrained params	# Total params
18-layer	11.4M	33.3M	(7, 2, 8, 1)	0.4M	31.7M
34-layer	21.5M	63.6M	(11, 4, 18, 1)	0.8M	60.5M
50-layer	23.9M	46.4M	(16, 6, 27, 1)	0.9M	45.8M
101-layer	42.8M	85.5M	(16, 6, 78, 1)	0.9M	84.8M
152-layer	58.5M	117.6M	(22, 12, 117, 1)	1.4M	115.6M

Table 6.6 Number of trainable parameters and configuration settings for DenseNet models of different depth.

DenseNet	2D-CNN	3D-CNN	Semi-CNN		
	# Params	# Params	Configuration	# Pretrained params	# Total params
121-layer	7.2M	11.4M	(25, 12, 80, 1)	0.8M	10.4M
169-layer	12.8M	18.8M	(37, 32, 96, 1)	0.8M	17.9M
201-layer	18.4M	25.6M	(25, 12, 160, 1)	0.8M	25.2M
161-layer	26.9M	39.7M	(25, 12, 120, 1)	1.7M	38.1M

6.4.3 Evaluation on validation dataset

This section presents the evaluation results for our architecture and the comparison of its performance with the corresponding 3D-CNN model. Since a previous study [122] has demonstrated that 3D-CNN outperforms 2D-CNN in action recognition tasks, we do not explicitly train 2D-CNN models for our comparison here.

The validation results on UCF-101 Split-1 dataset is presented in Table 6.7. The experimental results shown are obtained after training for 50 epochs, which might not have fully converged, but are presented here for comparison and reference. From the results in Table 6.7, we observe a significant improvement on the prediction accuracy for our semi-CNN when applied on

all the different models – VGG, ResNet and DenseNet. For the VGG-16 model, our architecture outperforms C3D [122] by 18% in accuracy. For ResNet models, an average of 16% increment is achieved, while for the Densenet-121 model, the boost is almost 30% accuracy. Deeper networks of ResNet and DenseNet models (such as ResNet-101, DenseNet-169 and DenseNet-201) are not presented here as the validation results deteriorate for both 3D- and semi-CNN architectures, demonstrating that the UCF-101 dataset is too small to be trained for a deep 3D network, as stated in [123, 141].

Table 6.7 Validation results on models VGG, ResNet and DenseNet for 3D-CNN and semi-CNN architectures comparison.

Model	# Layers	3D-CNN		Semi-CNN	
		Top-1 acc (%)	Top-5 acc (%)	Top-1 acc (%)	Top-5 acc (%)
VGG	16	36.53	62.70	54.27	81.21
ResNet	18	47.11	73.65	64.92	85.51
ResNet	34	48.19	74.41	66.53	88.58
ResNet	50	37.67	64.60	50.67	78.91
DenseNet	121	12.64	37.88	42.48	76.53

Figure 6.5 illustrates the comparison of the training performance for the models listed in Table 6.7. For 3D-CNN models, the gap between training and validation plot (blue and orange lines) is much bigger as compared to the plots in semi-CNN. This indicates that overfitting occurs when we train a more complex model (or a deeper network) on a relatively small dataset. Our architecture learns faster as can be seen in the steeper slope, which is a result of transfer learning which initializes the parameters in our spatial convolution layers. For deeper networks such as ResNet-50 and Densenet-121, huge fluctuations can be seen in the validation loss of the 3D architecture, which could mean that the network is overfitting as it could not generalize its training parameters on the validation set. Although fluctuations also occur in our semi-CNN architecture, it is more stable and the validation loss is progressively reduced when trained for longer epochs.

Model	Loss	
	3D-CNN	Semi-CNN
VGG-16		
ResNet-18		
ResNet-34		
ResNet-50		
DenseNet-121

Figure 6.5. Performance plots for 3D-CNN and semi-CNN architectures applied on different models – VGG, ResNet and DenseNet.

This empirical study provides strong support for our semi-CNN architecture in that using a fusion of convolution layers outperforms a full 3D convolution network, with additional advantages: 1) a lower number of training parameters, 2) transfer learning from pre-trained models is feasible, and 3) more stable training and faster convergence.

As there is insufficient data to enable the training of much deeper networks, two larger datasets for action recognition have been introduced – Kinetics dataset from DeepMind [140] with 400 action classes and 240k videos for training, and Sports1M [79] from Stanford and Google with 1 million YouTube videos and 487 action classes. Deeper 3D networks are commonly trained on these larger datasets before transfer learning to smaller datasets like UCF-101. Due to the limitation in computational resources, we only present our validation results on shallower networks on UCF-101 dataset, and we expect that the performance can be generalized to deeper networks when applied on larger datasets.

6.5 Features Visualization and Qualitative Results

Comparison

We report the comparisons of qualitative results for 3D- and Semi-ResNet-18 models, as well as illustrations of features for Semi-ResNet-18 in Figure 6.6. Top two rows of Figure 6.6 show 16 consecutive frames sampled from the validation video as network input. Subsequent three rows in Figure 6.6 illustrate examples of features extracted in the spatial, temporal and spatio-temporal space in Semi-ResNet-18 (as described in Table 6.2). Spatial features shown are obtained after the spatial convolution and pooling layers, with output dimension $(16 \times 14 \times 14)$. After the temporal convolution block and max pooling layer, we obtain temporal features with dimension $(8 \times 14 \times 14)$. The features are then processed by spatio-temporal blocks, where the output size is further reduced to $(2 \times 4 \times 4)$. At the bottom row, we illustrate the top-5 prediction scores for both 3D- and Semi-ResNet-18.

Figures 6.6 (a – c) show examples where our semi-CNN model outperforms 3D-ResNet, while Figures 6.6 (d, e) show examples of 3D-ResNet with better performance. Figure 6.6 (f) display example where both models make the correct prediction, while Figures 6.6 (g, h) display examples where both models fail. From the top-5 prediction scores, it is obvious that some of the predictions do not match with the action class. This could be due to: 1) the network has not fully converged and requires more training epoch, and 2) insufficient training data to train the high number of network parameters.

Figure 6.6. Examples of qualitative results for 3D- and Semi-ResNet-18 models.

6.6 Conclusion

This chapter demonstrates the effectiveness of our proposed architecture as compared to existing fusion models and 3D convolution models. We evaluated our framework on three popular networks – VGG, ResNet and DenseNet. Our empirical results show significant improvements over all the three networks when compared with its corresponding 3D convolution networks. In addition, our architecture shows faster convergence with the benefit of transfer learning, and has fewer trainable parameters which reduces overfitting issues. The learning properties for existing models are preserved as well as the network depth. More experiments can be conducted to find optimized configurations for the spatial, temporal and spatio-temporal convolution layers, as well as the aggregation of layers for each block. Deeper networks can be trained on larger datasets and fine-tune on UCF-101 dataset to further enhance the prediction accuracy.

CHAPTER 7

CONCLUSION

7.1 Research contributions

This thesis focuses on research in the areas of action recognition and human pose estimation based on monocular video inputs, with the ultimate goal of human motion analysis and reconstruction for real-life applications. The work exploits both deep learning and hand-crafted spatio-temporal features extracted from video frames to tackle the problems due to missing depth information. A spatio-temporal architecture is proposed and developed. It shows significant improvements over 3D convolution networks. Various investigations and experiments were conducted to discover baseline and extended configurations for our framework to allow effective learning and practical implementation. The contributions of each chapter are summarised as follows.

Chapter 3 starts with addressing the motion recovery task by implementing exemplar-based pose estimation for up to 12 consecutive frames. In contrast to conventional motion tracking approaches that require a stack of input frames to predict and update a single pose, our system can directly produce a sequence of smooth and realistic 3D poses for all the input frames at a time. Our method eliminates error propagation from previous frames and motion jittering during the reconstruction process. Occlusion and projection ambiguities are also resolved as we exploit motion and appearance feature coherence across consecutive frames.

From Chapter 4 onwards, we delve into developing an effective spatio-temporal convolutional architecture to address the action recognition task. We utilize transfer learning from pre-trained models to initialize our network and explore different configurations to study the model's complexity (the number of trainable parameters) and its performance. This chapter first

investigates the feasibility of a compact single-stream spatio-temporal architecture, by replacing the global average pooling layer in a 2D convolutional network with a 3D convolution and pooling layer. Our framework has a much lower number of parameters as compared to a full 3D model, and yet is able to encode high-level temporal features when compared to a full 2D model. A two-stream architecture requires computation of motion features such as optical flow for temporal encoding, but our model with a single-stream input can directly process video frames and encode temporal coherence using a 3D convolution layer. A weighted aggregation function is also introduced to improve discrimination of local motions from different actions. By employing a sparse sampling strategy, varying temporal dynamics can be encoded without affecting the input dimension and architecture structure. Validation results on the UCF-101 dataset demonstrate that our framework outperforms existing 3D architectures using both dense and sparse sampling strategies.

Chapter 5 further improves the work in Chapter 4 by extending the learning capacity of the single-stream spatio temporal network. The network performance is evaluated by applying additional 3D convolution layers, different global representations for features aggregation, and encoding of varying temporal dynamics in a pyramid representation. Building a deeper model helps to boost prediction accuracy, but at the cost of increased model complexity. Computing the global representation from the mean of all the local features is the most effective and cost-efficient method for implementing weighted features aggregation. Hierarchical encoding of temporal dynamics performs better than using a fixed temporal stride, without affecting the number of trainable parameters in the architecture. This chapter draws attention to the importance of tradeoff considerations between model complexity (computational cost) and prediction accuracy when developing a high-efficiency architecture for practical implementation.

With the learning capability and potential demonstrated in the single-stream spatio-temporal network, Chapter 6 generalizes the concept to existing

convolutional networks by introducing a variant of the spatio-temporal fusion method. Different from other fusion models in the literature, our architecture starts with spatial convolution at the lower level, followed by temporal convolution for intermediate features. At the higher level, spatio-temporal convolution is performed before connecting to a fully-connected layer for classification prediction. Our framework is able to retain individual network learning properties, such as residual learning in ResNet and dense connectivity in DenseNet, while preserving the same convolution configurations for our temporal and spatio-temporal blocks. To retain the same network depth as its corresponding 2D model, we split the layers such that the top part forms 3D convolution layers, while the middle layers are transformed to 1D convolution layers. Our validation results show significant boosts in performance with 16 – 30% increments when compared with the corresponding 3D models. This empirical study provides valuable insights on the effective learning of spatio-temporal features by decomposition into subspace learning – spatial and temporal convolutions at the lower layers before encoding high-level spatio-temporal features. Aggregation of features learning in the spatial and temporal space not only reduces the number of trainable parameters, but also reduces overfitting problems and allows transfer learning from pre-trained models.

We summarize the contributions of this thesis into four main points:

- 1) Addressed human motion recovery with a direct motion reconstruction method that estimates a sequence of smooth poses from a stack of frames, outperforming prior tracking methods.
- 2) A compact and flexible spatio-temporal network with weighted global aggregation is proposed with much lower computational cost and faster convergence than existing 3D models. More complicated models can be built and extended from this compact network to boost its accuracy.
- 3) A semi-CNN architecture with fusion of 1D, 2D, and 3D convolution layers is proposed and adopted on three popular networks – VGG, ResNet and DenseNet. Evaluation results on semi-CNN show significant boosts in

performance with 16 – 30% increments when compared with their corresponding 3D models.

4) Valuable insights were obtained during the experimental study on the effective learning of spatio-temporal features. Decomposition of features into subspace learning – spatial and temporal convolutions at the lower layers before high-level spatio-temporal encoding, not only reducing overfitting problems with lower parameters, but also improve prediction accuracy significantly with faster and more stable convergence.

7.2 Directions for future work

This section discusses the limitations of the current work and makes recommendations for future work. We first look into the short-term directions for the two main processes: 1) human motion recovery and 2) action recognition. Then, a long-term direction is suggested to map out the pipeline of a human motion analysis system.

Recent works on human motion recovery with deep learning have reported real-time reconstruction with comparable results to RGBD images, as discussed in Section 2.1.3.1. Existing works perform pose estimation from a single frame or a stack of consecutive frames. Meanwhile, our work suggests a new direction for direct motion reconstruction from a sequence of frames. Our current implementation is limited to smaller datasets where the motion backgrounds are almost static. We require pre-processing to re-aligning the subject before extracting hand-crafted features to form dense video patch descriptors. To improve on the state of the art, we propose to utilize deep learning to learn the mapping between a stack of consecutive images (with unconstrained background) to a sequence of continuous 3D poses. However, the network training would be challenging as the number of parameters would be extremely large for prediction of several poses at the same time. Besides, much larger datasets are required to train the network such that the training process can converge and to avoid overfitting problems. With careful

design of the architecture together with the availability of powerful computational resources, deep learning motion recovery could be further explored.

Action recognition with deep learning has advances with numerous studies on convolutional networks and recurrent networks. Training on large-scale datasets and then transfer learning to smaller datasets demonstrated superior performance with boosted prediction accuracy and faster training. Our empirical study on developing an effective spatio-temporal framework has reported outstanding performance when compared to its corresponding 3D convolutional networks and prior fusion approaches. The current implementation can be extended to deeper models and training on larger datasets, such that we can fully investigate the learning capacity of our architecture, also to allow fair comparison with the state-of-the-art approaches. In future, we can generalize our semi-CNN framework on other networks (such as Inception, VGG with batch normalization), as well as finding optimized configurations for the architecture within practical computational cost. Exploring discriminative local motions from different actions can be further investigated as this provides insights on the underlying spatio-temporal features, which will be extremely useful when applied in fine-grained action recognition.

In the longer-term roadmap, developing a full pipeline for human motion analysis in specific application, especially in the field of healthcare and sport science, would greatly benefit the community as it provides more accurate analysis without requiring any equipment setup or skilled manpower. As this thesis has covered the system main processes – human pose estimation, motion reconstruction and action recognition, the next step is to process and analyze the output of the system together with the subject's information. With the personalization feature, patients or athletes could track their progress and compare their performance with others. Our pipeline can also be applied to general applications such as surveillance detection to track anomalous actions, or motion retargeting to animate 3D models.

7.3 Conclusion

In conclusion, the work reported in this thesis has met all the objectives and scopes set to investigate the processes in human motion analysis. For human motion recovery task, we addressed the problem with a direct motion reconstruction method that can estimate a sequence of poses from a stack of frames. We exploited example-based methods with hand-crafted spatio-temporal features to reconstruct smooth and realistic motions. Our method outperforms prior stochastic sampling tracking methods and achieved comparative performance with state-of-the-art learning-based methods.

For action recognition tasks, we proposed a compact architecture with a single-stream spatio-temporal convolution network. Transfer learning was adopted for improved training and prediction accuracy. Different configurations and experiments have been conducted to validate the learning capability of our architecture with a practical computational cost. Empirical study shows comparable performance with 3D convolutional networks, but with a much lower complexity. We further extended and generalized the concept to develop a semi-CNN framework that can be applied to existing networks without affecting its learning properties. Evaluations on VGG, ResNet and DenseNet models demonstrated superior performance when compared with its corresponding 3D convolutional networks and prior fusion approach. Besides, our architecture also shows advantages in lower trainable parameters, reduced overfitting issues and stable training. The framework can be further extended to deeper models and training on larger datasets.

With the process investigation and architecture development, this thesis fulfilled the motivations behind the research intention, which aims to address the limitations in the traditional motion capture approaches. Our system is not constrained to be used in any specific environment and does not require specific devices to capture and process the motion data. It is practical for implementation and easy to operate. This is promising for the

development of a complete system for human motion analysis that could be realized in the near future.

REFERENCE

1. Moeslund, T.B., A. Hilton, and V. Krüger, *A survey of advances in vision-based human motion capture and analysis*. Computer vision and image understanding, 2006. 104(2): p. 90-126.
2. Hen, Y.W. and R. Paramesran. *Single camera 3d human pose estimation: A review of current techniques*. in *Technical Postgraduates (TECHPOS), 2009 International Conference for*. 2009. IEEE.
3. Perez-Sala, X., et al., *A survey on model based approaches for 2D and 3D visual human pose recovery*. Sensors (Basel), 2014. 14(3): p. 4189-210.
4. LeCun, Y., et al., *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 1998. 86(11): p. 2278-2324.
5. Ji, X. and H. Liu, *Advances in view-invariant human motion analysis: A review*. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 2010. 40(1): p. 13-24.
6. Aggarwal, J.K. and Q. Cai, *Human motion analysis: A review*. Computer vision and image understanding, 1999. 73(3): p. 428-440.
7. Moeslund, T.B. and E. Granum, *A survey of computer vision-based human motion capture*. Computer vision and image understanding, 2001. 81(3): p. 231-268.
8. Wang, L., W. Hu, and T. Tan, *Recent developments in human motion analysis*. Pattern recognition, 2003. 36(3): p. 585-601.
9. Weinland, D., R. Ronfard, and E. Boyer, *A survey of vision-based methods for action representation, segmentation and recognition*. Computer Vision and Image Understanding, 2011. 115(2): p. 224-241.
10. Zhou, F. and F. De la Torre, *Spatio-temporal Matching for Human Pose Estimation in Video*. 2016.
11. Sigal, L. and M.J. Black, *Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion*. Brown University TR, 2006. 120.
12. *CMU Graphics Lab Motion Capture Database*. Available from: <http://mocap.cs.cmu.edu/>.
13. Soomro, K., A.R. Zamir, and M. Shah, *UCF101: A dataset of 101 human actions classes from videos in the wild*. arXiv preprint arXiv:1212.0402, 2012.
14. Kuehne, H., et al. *HMDB: a large video database for human motion recognition*. in *Computer Vision (ICCV), 2011 IEEE International Conference on*. 2011. IEEE.
15. Liu, Z., et al., *A survey of human pose estimation: the body parts parsing based methods*. Journal of Visual Communication and Image Representation, 2015. 32: p. 10-19.
16. Zhang, H.-B., et al., *A Survey on Human Pose Estimation*. Intelligent Automation & Soft Computing, 2015: p. 1-7.
17. Nishani, E. and B. Çiço. *Computer vision approaches based on deep learning and neural networks: Deep neural networks for video analysis of human pose estimation*. in *Embedded Computing (MECO), 2017 6th Mediterranean Conference on*. 2017. IEEE.
18. Poppe, R., *Vision-based human motion analysis: An overview*. Computer vision and image understanding, 2007. 108(1): p. 4-18.
19. Ramanan, D., *Part-based models for finding people and estimating their pose*, in *Visual Analysis of Humans*. 2011, Springer. p. 199-223.
20. Sminchisescu, C., *3D human motion analysis in monocular video: techniques and challenges*, in *Human Motion*. 2008, Springer. p. 185-211.

21. Felzenszwalb, P., D. McAllester, and D. Ramanan. *A discriminatively trained, multiscale, deformable part model*. in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. 2008. IEEE.
22. Fischler, M.A. and R.A. Elschlager, *The representation and matching of pictorial structures*. IEEE Transactions on computers, 1973. 22(1): p. 67-92.
23. Felzenszwalb, P.F. and D.P. Huttenlocher, *Pictorial structures for object recognition*. International Journal of Computer Vision, 2005. 61(1): p. 55-79.
24. Andriluka, M., S. Roth, and B. Schiele. *Pictorial structures revisited: People detection and articulated pose estimation*. in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. 2009. IEEE.
25. Belongie, S., J. Malik, and J. Puzicha, *Shape matching and object recognition using shape contexts*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2002. 24(4): p. 509-522.
26. Freund, Y. and R.E. Schapire. *A decision-theoretic generalization of on-line learning and an application to boosting*. in *Computational learning theory*. 1995. Springer.
27. Wang, Y. and G. Mori, *Multiple tree models for occlusion and spatial constraints in human pose estimation*, in *Computer Vision-ECCV 2008*. 2008, Springer. p. 710-724.
28. Eichner, M. and V. Ferrari, *We are family: Joint pose estimation of multiple persons*. Computer Vision-ECCV 2010, 2010: p. 228-242.
29. Sapp, B., A. Toshev, and B. Taskar, *Cascaded models for articulated pose estimation*. Computer Vision-ECCV 2010, 2010: p. 406-420.
30. Dantone, M., et al. *Human pose estimation using body parts dependent joint regressors*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013.
31. Dantone, M., et al., *Body parts dependent joint regressors for human pose estimation in still images*. IEEE transactions on pattern analysis and machine intelligence, 2014. 36(11): p. 2131-2143.
32. Ladicky, L., P.H. Torr, and A. Zisserman. *Human pose estimation using a joint pixel-wise and part-wise formulation*. in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013.
33. Dalal, N. and B. Triggs. *Histograms of oriented gradients for human detection*. in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. 2005. IEEE.
34. Yang, Y. and D. Ramanan. *Articulated pose estimation with flexible mixtures-of-parts*. in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. 2011. IEEE.
35. Yang, Y. and D. Ramanan, *Articulated human detection with flexible mixtures of parts*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2013. 35(12): p. 2878-2890.
36. Bourdev, L. and J. Malik. *Poselets: Body part detectors trained using 3d human pose annotations*. in *Computer Vision, 2009 IEEE 12th International Conference on*. 2009. IEEE.
37. Wang, Y., D. Tran, and Z. Liao. *Learning hierarchical poselets for human parsing*. in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. 2011. IEEE.
38. Pishchulin, L., et al. *Poselet conditioned pictorial structures*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013.
39. Pirsiaavash, H. and D. Ramanan. *Steerable part models*. in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. 2012. IEEE.

40. Felzenszwalb, P.F., et al., *Object detection with discriminatively trained part-based models*. IEEE transactions on pattern analysis and machine intelligence, 2010. 32(9): p. 1627-1645.
41. Pons-Moll, G., D.J. Fleet, and B. Rosenhahn. *Posebits for monocular human pose estimation*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014.
42. Shakhnarovich, G., P. Viola, and T. Darrell. *Fast pose estimation with parameter-sensitive hashing*. in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. 2003. IEEE.
43. Poppe, R., *Evaluating example-based pose estimation: Experiments on the humaneva sets*. 2007.
44. Fragkiadaki, K., H. Hu, and J. Shi. *Pose from flow and flow from pose*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013.
45. Liu, J., et al., *3D Human motion tracking by exemplar-based conditional particle filter*. Signal Processing, 2015. 110: p. 164-177.
46. Zhou, X. and X. Li, *Dynamic spatio-temporal modeling for example-based human silhouette recovery*. Signal Processing, 2015. 110: p. 27-36.
47. Agarwal, A. and B. Triggs, *Recovering 3D human pose from monocular images*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2006. 28(1): p. 44-58.
48. Hara, K. and R. Chellappa. *Computationally efficient regression on a dependency graph for human pose estimation*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013.
49. Simo-Serra, E., et al. *Single image 3D human pose estimation from noisy observations*. in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. 2012. IEEE.
50. Rothrock, B., S. Park, and S.-C. Zhu. *Integrating grammar and segmentation for human pose estimation*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013.
51. Jain, A., et al., *Learning human pose estimation features with convolutional networks*. arXiv preprint arXiv:1312.7302, 2013.
52. Li, S. and A.B. Chan. *3d human pose estimation from monocular images with deep convolutional neural network*. in *Asian Conference on Computer Vision*. 2014. Springer.
53. Li, S., Z.-Q. Liu, and A.B. Chan. *Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014.
54. Tompson, J.J., et al. *Joint training of a convolutional network and a graphical model for human pose estimation*. in *Advances in neural information processing systems*. 2014.
55. Chen, X. and A.L. Yuille. *Articulated pose estimation by a graphical model with image dependent pairwise relations*. in *Advances in Neural Information Processing Systems*. 2014.
56. Toshev, A. and C. Szegedy. *Deeppose: Human pose estimation via deep neural networks*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014.
57. Pfister, T., J. Charles, and A. Zisserman. *Flowing convnets for human pose estimation in videos*. in *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

58. Newell, A., K. Yang, and J. Deng. *Stacked hourglass networks for human pose estimation*. in *European Conference on Computer Vision*. 2016. Springer.
59. Mehta, D., et al., *VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera*. arXiv preprint arXiv:1705.01583, 2017.
60. Fragkiadaki, K., et al. *Recurrent network models for human dynamics*. in *Proceedings of the IEEE International Conference on Computer Vision*. 2015.
61. Belagiannis, V. and A. Zisserman, *Recurrent human pose estimation*. arXiv preprint arXiv:1605.02914, 2016.
62. Lawrence, N.D. *Gaussian process latent variable models for visualisation of high dimensional data*. in *Advances in neural information processing systems*. 2004.
63. Ek, C.H., P.H. Torr, and N.D. Lawrence. *Gaussian process latent variable models for human pose estimation*. in *International workshop on machine learning for multimodal interaction*. 2007. Springer.
64. Andriluka, M., S. Roth, and B. Schiele. *People-tracking-by-detection and people-detection-by-tracking*. in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. 2008. IEEE.
65. Lawrence, N.D. and A.J. Moore. *Hierarchical Gaussian process latent variable models*. in *Proceedings of the 24th international conference on Machine learning*. 2007. ACM.
66. Andriluka, M., S. Roth, and B. Schiele. *Monocular 3d pose estimation and tracking by detection*. in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. 2010. IEEE.
67. Ferrari, V., M. Marin-Jimenez, and A. Zisserman. *Progressive search space reduction for human pose estimation*. in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. 2008. IEEE.
68. Eichner, M., et al., *2d articulated human pose estimation and retrieval in (almost) unconstrained still images*. *International Journal of Computer Vision*, 2012. 99(2): p. 190-214.
69. Elgammal, A. and C.-S. Lee, *Tracking people on a torus*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009. 31(3): p. 520-538.
70. Bo, L. and C. Sminchisescu, *Twin gaussian processes for structured prediction*. *International Journal of Computer Vision*, 2010. 87(1): p. 28-52.
71. Taylor, G.W., et al. *Dynamical binary latent variable models for 3d human pose tracking*. in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. 2010. IEEE.
72. Taylor, G.W., G.E. Hinton, and S.T. Roweis. *Modeling human motion using binary latent variables*. in *Advances in neural information processing systems*. 2007.
73. Yao, B. and L. Fei-Fei. *Modeling mutual context of object and human pose in human-object interaction activities*. in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. 2010. IEEE.
74. Eichner, M. and V. Ferrari, *Human pose co-estimation and applications*. *IEEE transactions on pattern analysis and machine intelligence*, 2012. 34(11): p. 2282-2288.
75. Abbasi, M., H.R. Rabiee, and C. Gagné. *Monocular 3D human pose estimation with a semi-supervised graph-based method*. in *3D Vision (3DV), 2015 International Conference on*. 2015. IEEE.
76. Li, H., et al., *Automatic detection and recognition of athlete actions in diving video*, in *Advances in Multimedia Modeling*. 2006, Springer. p. 73-82.

77. Ji, S., et al., *3D convolutional neural networks for human action recognition*. IEEE transactions on pattern analysis and machine intelligence, 2013. 35(1): p. 221-231.
78. Jung, M., J. Hwang, and J. Tani. *Multiple spatio-temporal scales neural network for contextual visual recognition of human actions*. in *Development and Learning and Epigenetic Robotics (ICDL-Epirob), 2014 Joint IEEE International Conferences on*. 2014. IEEE.
79. Karpathy, A., et al. *Large-scale video classification with convolutional neural networks*. in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014.
80. Sun, L., et al. *Human action recognition using factorized spatio-temporal convolutional networks*. in *Proceedings of the IEEE International Conference on Computer Vision*. 2015.
81. Ng, J.Y.-H., et al., *Actionflownet: Learning motion representation for action recognition*. arXiv preprint arXiv:1612.03052, 2016.
82. Simonyan, K. and A. Zisserman. *Two-stream convolutional networks for action recognition in videos*. in *Advances in neural information processing systems*. 2014.
83. Wang, L., Y. Qiao, and X. Tang. *Action recognition with trajectory-pooled deep-convolutional descriptors*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
84. Wang, H. and C. Schmid. *Action recognition with improved trajectories*. in *Computer Vision (ICCV), 2013 IEEE International Conference on*. 2013. IEEE.
85. Wang, Z., et al. *Exploring fisher vector and deep networks for action spotting*. in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2015 IEEE Conference on*. 2015. IEEE.
86. Gkioxari, G. and J. Malik. *Finding action tubes*. in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. 2015. IEEE.
87. Tu, Z., et al., *Multi-stream CNN: Learning representations based on human-related regions for action recognition*. Pattern Recognition, 2018. 79: p. 32-43.
88. Zhang, B., et al. *Real-time action recognition with enhanced motion vector CNNs*. in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. 2016. IEEE.
89. Sun, S., et al. *Optical flow guided feature: A fast and robust motion representation for video action recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
90. Diba, A., V. Sharma, and L. Van Gool. *Deep temporal linear encoding networks*. in *Computer Vision and Pattern Recognition*. 2017.
91. Girdhar, R., et al. *ActionVLAD: Learning spatio-temporal aggregation for action classification*. in *CVPR*. 2017.
92. Feichtenhofer, C., A. Pinz, and A. Zisserman. *Convolutional two-stream network fusion for video action recognition*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.
93. Huang, C.-D., C.-Y. Wang, and J.-C. Wang. *Human action recognition system for elderly and children care using three stream ConvNet*. in *Orange Technologies (ICOT), 2015 International Conference on*. 2015. IEEE.
94. Chéron, G., I. Laptev, and C. Schmid. *P-cnn: Pose-based cnn features for action recognition*. in *Proceedings of the IEEE international conference on computer vision*. 2015.

95. Tu, Z., et al. *MSR-CNN: applying motion salient region based descriptors for action recognition*. in *Pattern Recognition (ICPR), 2016 23rd International Conference on*. 2016. IEEE.
96. Cao, C., et al., *Body Joint guided 3D Deep Convolutional Descriptors for Action Recognition*. arXiv preprint arXiv:1704.07160, 2017.
97. Choutas, V., et al. *Potion: Pose motion representation for action recognition*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
98. Baccouche, M., et al. *Sequential deep learning for human action recognition*. in *International Workshop on Human Behavior Understanding*. 2011. Springer.
99. Gers, F.A., N.N. Schraudolph, and J. Schmidhuber, *Learning precise timing with LSTM recurrent networks*. *Journal of machine learning research*, 2002. 3(Aug): p. 115-143.
100. Hu, S.-H., Y. Li, and B. Li. *Video2vec: Learning semantic spatio-temporal embeddings for video representation*. in *Pattern Recognition (ICPR), 2016 23rd International Conference on*. 2016. IEEE.
101. Cho, K., et al., *On the properties of neural machine translation: Encoder-decoder approaches*. arXiv preprint arXiv:1409.1259, 2014.
102. Singh, B., et al. *A multi-stream bi-directional recurrent neural network for fine-grained action detection*. in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. 2016. IEEE.
103. Ng, J.Y.-H., et al. *Beyond short snippets: Deep networks for video classification*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
104. Hou, J., et al. *Multimedia event detection via deep spatial-temporal neural networks*. in *Multimedia and Expo (ICME), 2016 IEEE International Conference on*. 2016. IEEE.
105. Srivastava, N., E. Mansimov, and R. Salakhudinov. *Unsupervised learning of video representations using lstms*. in *International conference on machine learning*. 2015.
106. Hinton, G.E. and R.R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*. *science*, 2006. 313(5786): p. 504-507.
107. Misra, I., C.L. Zitnick, and M. Hebert. *Shuffle and learn: unsupervised learning using temporal order verification*. in *European Conference on Computer Vision*. 2016. Springer.
108. Dalal, N., B. Triggs, and C. Schmid. *Human detection using oriented histograms of flow and appearance*. in *European conference on computer vision*. 2006. Springer.
109. Uijlings, J., et al., *Video classification with densely extracted hog/hof/mbh features: an evaluation of the accuracy/computational efficiency trade-off*. *International Journal of Multimedia Information Retrieval*, 2015. 4(1): p. 33-44.
110. Wang, H., et al., *Dense trajectories and motion boundary descriptors for action recognition*. *International journal of computer vision*, 2013. 103(1): p. 60-79.
111. Kantorov, V. and I. Laptev. *Efficient feature extraction, encoding and classification for action recognition*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014.
112. Tekin, B., et al. *Direct prediction of 3D body poses from motion compensated sequences*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.
113. Akilan, T., Q.J. Wu, and Y. Yang, *Fusion-based foreground enhancement for background subtraction using multivariate multi-model Gaussian distribution*. *Information Sciences*, 2018. 430: p. 414-431.

114. McKenna, S.J., et al. *Tracking interacting people*. in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*. 2000. IEEE.
115. Bouwmans, T., et al., *Background Modeling and Foreground Detection for Video Surveillance*. 2014: Chapman and Hall/CRC.
116. Müller, M., *Information retrieval for music and motion*. Vol. 2. 2007: Springer.
117. Agarwal, A. and B. Triggs. *3D human pose from silhouettes by relevance vector regression*. in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. 2004. IEEE.
118. Wang, C., et al. *Robust estimation of 3d human poses from a single image*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014.
119. Sigal, L., et al., *Loose-limbed people: Estimating 3d human pose and motion using non-parametric belief propagation*. *International journal of computer vision*, 2012. 98(1): p. 15-48.
120. Simo-Serra, E., et al. *A joint model for 2D and 3D pose estimation from a single image*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013.
121. Zhou, X., et al., *MonoCap: Monocular Human Motion Capture using a CNN Coupled with a Geometric Prior*. arXiv preprint arXiv:1701.02354, 2017.
122. Tran, D., et al. *Learning spatiotemporal features with 3d convolutional networks*. in *Computer Vision (ICCV), 2015 IEEE International Conference on*. 2015. IEEE.
123. Carreira, J. and A. Zisserman. *Quo vadis, action recognition? a new model and the kinetics dataset*. in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. 2017. IEEE.
124. Varol, G., I. Laptev, and C. Schmid, *Long-term temporal convolutions for action recognition*. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 40(6): p. 1510-1517.
125. Wang, L., et al. *Temporal segment networks: Towards good practices for deep action recognition*. in *European Conference on Computer Vision*. 2016. Springer.
126. Li, Y., et al. *Vlad3: Encoding dynamics of deep features for action recognition*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.
127. Bilen, H., et al. *Dynamic image networks for action recognition*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.
128. Zhu, J., W. Zou, and Z. Zhu, *End-to-end Video-Level Representation Learning for Action Recognition*. arXiv preprint arXiv:1711.04161, 2017.
129. Deng, J., et al., *Imagenet: A large-scale hierarchical image database*. 2009.
130. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
131. Ioffe, S. and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. arXiv preprint arXiv:1502.03167, 2015.
132. Paszke, A., et al., *Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration*. 2017.
133. Huang, G., et al. *Densely connected convolutional networks*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
134. Diba, A., et al., *Temporal 3D ConvNets: New Architecture and Transfer Learning for Video Classification*. arXiv preprint arXiv:1711.08200, 2017.
135. Wang, L., et al., *Temporal segment networks for action recognition in videos*. *IEEE transactions on pattern analysis and machine intelligence*, 2018.
136. Simonyan, K. and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*. arXiv preprint arXiv:1409.1556, 2014.

- 137. Tran, D., et al. *A closer look at spatiotemporal convolutions for action recognition*. in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018.
- 138. Yao, T. and X. Li, *Yh technologies at activitynet challenge 2018*. arXiv preprint arXiv:1807.00686, 2018.
- 139. Hara, K., H. Kataoka, and Y. Satoh. *Learning spatio-temporal features with 3D residual networks for action recognition*. in *Proceedings of the ICCV Workshop on Action, Gesture, and Emotion Recognition*. 2017.
- 140. Kay, W., et al., *The kinetics human action video dataset*. arXiv preprint arXiv:1705.06950, 2017.
- 141. Hara, K., H. Kataoka, and Y. Satoh. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA. 2018.