Online Spatiotemporal Action Detection and Prediction via Causal Representations

Gurkirt Singh

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Mathematics

Visual Artificial Intelligence Lab
School of Engineering, Computing and Mathematics
Oxford Brookes University

Supervised by :
Prof. Fabio Cuzzolin and Prof. Nigel Crook

September 2019

The author holds the **copyright** *of this thesis. Any person(s) intending to use a part or whole of the materials in the thesis in a proposed publication must seek copyright release from the author.*

Thesis/Assessment Committee

Professor Andrea Vedaldi (External Examiner)
Visual Geometry Group (VGG)
University of Oxford

Professor Fridolin Wild (Internal Examiner)
School of Engineering, Computing and Mathematics
Oxford Brookes University

Preface

This dissertation is submitted for the degree of Doctor of Philosophy at The Oxford Brookes University of United Kingdom. The research presented herein was undertaken under the kind supervision of Prof. Fabio Cuzzolin and Prof. Nigel Crook between September 2015 and August 2019. To the best of my knowledge, this work is original, excluding those previous works for which acknowledgement and reference have been made. Neither this, nor any substantially similar dissertation has been or is being submitted for any other degree, diploma or qualification at any other university. Part of this work are published in the following publications:

- **Gurkirt Singh** and Fabio Cuzzolin, "Recurrent Convolutions for Causal 3D CNNs", in Proceedings of International Conference on Computer Vision Workshop (ICCVW) on Large Scale Holistic Video Understanding, 2019.
- **Gurkirt Singh**, Suman Saha and Fabio Cuzzolin, "TraMNet - Transition Matrix Network for Efficient Action Tube Proposals", in Proceedings of Asian Conference on Computer Vision (ACCV), 2018.
- **Gurkirt Singh**, Suman Saha and Fabio Cuzzolin, "Predicting Action Tubes", in Proceedings of European Conference on Computer Vision Workshop (ECCVW) on Anticipating Human Behaviour, 2018.
- **Gurkirt Singh**, Suman Saha, and Fabio Cuzzolin, "Online Real-time Multiple Spatio-temporal Action Localisation and Prediction", in proceedings of International Conference on Computer Vision (ICCV) 2017.
- Suman Saha, **Gurkirt Singh** and Fabio Cuzzolin, "AMTnet: Action-Micro-Tube Regression by end-to-end Trainable Deep Architecture", in proceedings of International Conference on Computer Vision (ICCV) 2017.
- Suman Saha, **Gurkirt Singh**, Michael Sapienza, Philip Torr and Fabio Cuzzolin, Deep Learning for Detecting Multiple Space-time Action Tubes in Videos, proceedings of British Machine Vision Conference (BMVC) 2016.
- **Gurkirt Singh** and Fabio Cuzzolin, "Untrimmed Video Classification for Activity Detection: Submission to ActivityNet Challenge", in Tech-report arXiv:1607.01979, presented at ActivityNet Challenge Workshop in CVPR 2016.

To my parents
And
my wife

Without whom none of my success would be possible.

Acknowledgements

First and foremost I would like to express my heartfelt thanks to my director of studies Prof. Fabio Cuzzolin, and co-supervisor Prof. Nigel Crook for providing such an exceptional opportunity for me to pursue a doctorate in such an interesting topic as Computer Vision! I also thank my supervisors for their invaluable time, guidance and support. In particular, Fabio's strive for excellence, openness and positive criticism always inspired me to push myself beyond my limits and allowed me to gradually improve my scientific, technical and academic skills through several projects. Moreover, human values like elegance and generosity that I found in Prof. Cuzzolin have inspired me to work in the Visual Artificial Intelligence Lab, School of Engineering, Computing and Mathematics, Oxford Brookes University. Furthermore, I am grateful to Professor Philip H. S. Torr for allowing me to work in a close collaboration with the world-renowned Torr Vision Group (TVG), the Department of Engineering Science, University of Oxford.

I would also like to thank my co-author Dr. Michael Sapienza for his time and effort during our long technical discussions. A special thanks to my labmate and co-author Suman Saha for his close collaboration and daily discussions which led to several exciting works and many publications.

I want to thank: Tjeerd Olde Scheper for his invaluable guidance and support as research tutor; Jill Organ, Catherine Joyejob, Lynn Farrell who helped me in all administrative tasks; Gerald Roy for excellent technical, software and hardware support; students at Brookes with whom I had a chance to work, especially Stephen Akrigg, Valentina Fontana, Kurt Degiorgio, Manuele Di Maio, Shantanu Rathod. I am grateful to all my colleagues and friends I met over the past few years: Mohamed Idries, Bedour Alshaigy, Mireya Munoz Balbontin, Alla Vovk, Will Guest, Jalawi Alshuduki, Cristian Roman at the TDE department, School of Engineering, Computing and Mathematics, Oxford Brookes University.

My special thanks go to Prof. Leonid Sigal for offering me an internship at Disney Research Pittsburgh (DRP), USA. Many thanks to Leonid and Andreas Lehrmann for their invaluable time and guidance during those my stay at DRP lab. I would also like to thank Prof. Greg Mori and Prof. Leonid Sigal for providing me with the opportunity to do another internship at BorealisAI, Canada. I feel fortunate to have had the chance to work in a highly stimulating environment at DRP and BorealisAI. Special thanks go to my DRP and BorealisAI colleagues and friends for their help and support Andreas Lehrmann,

Rajitha Navarantha, Ziad Al-Halah, Hongsuck Seo, Judith Butepage, He Jiawei, Suhail Mohamad, Thibaut Durand, Mengyao Zhai, Lei Chen, Gabrial, Lilli meng, Huyen Mori.

Due acknowledgement goes to my MSc supervisors Prof. Radu Horaud and Dr. Georgios Evangelidis for their time, advice and support during my MSc dissertation, and for providing me the opportunity to work as a research intern at INRIA, Grenoble. My acknowledgement would not be complete without mentioning a few names of my colleagues I worked with for the past few years at Siemens, Bangalore, India. I am thankful to my Siemens colleagues for teaching me so many things: Parmeet Bhatia, Prabhu Teja, Yogesh, Todd Wegner. My heartfelt thanks go to Dr. Amit Kale for his guidance throughout my time at Siemens.

I would also like to thank my friends, who have greatly supported me for many years, Rajjan Singh Thakur, Vijendra Kumar, Abhishek Kumar, Rajesh Ranjan, Prince Sharma, Mohamed Idries, Noemi Dreksler, Mireya Munoz Balbontin, Bedour Alshaigy, Rishab Mehta, Rajvee Mehta.

Last but not least, I am thankful for my *parents* and my *family* in India for their unending patience, encouragement and generosity. Especially, my parent for fighting for my life during the initial ten years of my life and those countless hospital visits; my sister *Mandeep Kaur* for homeschooling me during those hard times. Without their perseverance, I would not be writing this today. My heartfelt thanks go to my wife *Meenakshi* for her unbounded patience and pushing me to be my best self. She has been a rock in my life during hard times of late night paper submissions. She enjoyed my success as her own, she was always there. I can not thank her enough for being so supportive during this journey.

Abstract

In this thesis, we focus on video action understanding problems from an online and real-time processing point of view. We start with the conversion of the traditional offline spatiotemporal action detection pipeline into an online spatiotemporal action tube detection system. An action tube is a set of bounding connected over time, which bounds an action instance in space and time. Next, we explore the future prediction capabilities of such detection methods by extending the an existing action tube into the future by regression. Later, we seek to establish that online/causal representations can achieve similar performance to that of offline three dimensional (3D) convolutional neural networks (CNNs) on various tasks, including action recognition, temporal action segmentation and early prediction.

To this end, we propose various action tube detection approaches from either single or multiple frames. We start by introducing supervised action proposals for frame-level action detection and solving two energy optimisation formulations to detect the spatial and temporal boundaries of action tubes. Further, we propose an incremental tube construction algorithm to handle the online action detection problem. There, the real-time capabilities are made possible by introducing real-time frame-level action detection and real-time optical flow in the action detection pipeline for efficiency. Next, we extend our frame-level approach to multiple frames with the help of a novel proposal to for predicting flexible action 'micro-tubes' from a pair of frames. We extend the micro-tube prediction network in order to regress the future of each micro-tube, which is then fed to our proposed future action tube prediction framework. We convert 3D CNNs to causal 3D CNNs by replacing every 3D convolution with recurrent convolution, and by making use of sophisticated initialisation to handle the problems of recurrent modules.

We show that our action tube detectors perform better than previous state-of-the-art methods, while exhibiting online and real-time capabilities. We evaluate each action tube detector and predictor on publicly available benchmarks to show the comparison with other state-of-the-art approaches. We also show that our flexible micro-tube proposals not only improve action detection performance but can also handle sparse annotations. Finally, we demonstrate the causal capabilities of our causal 3D CNN.

Contents

Preface vii

Dedication ix

Acknowledgements xi

Abstract xiii

List of Figures xxi

List of Tables xxiii

Nomenclature xxv

1 Introduction 1

1.1 Understanding video content 1
1.2 Why human actions? 2
1.3 Types of problems in human action understanding 3
1.4 Why online action tube detection 4
1.5 Contributions of the thesis 6
1.6 Overview of the thesis 7
1.7 Structure of the chapters 8
1.8 Resulting publications and resources 9
1.8.1 List of publications 9
1.8.2 List of software packages and other resources 10

2 Related Work 13

2.1 Action recognition 13
2.1.1 Traditional approaches 13
2.1.2 2D CNNs 14
2.1.3 3D CNNs 14
2.2 Temporal action detection 15
2.3 Multi-label temporal action segmentation 15

2.4 Spatiotemporal action detection 16
2.4.1 Single-frame based methods 16
2.4.2 Multi-frame based methods 17
2.4.3 3D representations for action detection 18
2.5 Online and real-time methods 18
2.6 Early action prediction and detection 18
2.7 Future prediction problems 19
2.8 Causal representations 19
2.9 Related datasets 20
2.9.1 UCF101-24 20
2.9.2 ActivityNet 20
2.9.3 J-HMDB-21 21
2.9.4 LIRIS-HARL 21
2.9.5 DALY 21
2.9.6 Kinetics 22
2.9.7 MultiThumos 22
2.10 Evaluation metrics 22
2.10.1 Classification accuracy 22
2.10.2 Mean average precision (mAP) 22
2.10.3 Completion-mAP 23
2.10.4 Prediction-mAP 23

Part I : Towards Online Action Detection **25**

3 Deep learning for Spatiotemporal Detection of Multiple Action Tubes **27**

3.1 Introduction 27
3.1.1 Action tube detection approach 28
3.1.2 Temporal action detection approach 29
3.2 Overview of action tube detection approach 30
3.2.1 Resulting contributions: 31
3.3 Frame-level detection framework 32
3.3.1 Region proposal network 32
3.3.2 Detection network 33
3.3.3 CNN training strategy 33
3.3.4 Fusion of appearance and motion cues 33
3.4 Building action paths. 34
3.4.1 Problem formulation 34
3.4.2 Optimisation formulation 35

3.5 Temporal action detection . . . 36
3.5.1 Frame-level temporal action detection for the ActivityNet challenge . . . 36
3.5.2 Temporal trimming in action paths . . . 37
3.6 Experimental validation . . . 38
3.6.1 Temporal detection performance comparison . . . 38
3.6.2 Spatiotemporal detection performance comparison . . . 39
3.6.3 Comparative analysis of region proposal quality . . . 41
3.6.4 Impact of label smoothing . . . 42
3.6.5 Computing time analysis for testing . . . 44
3.6.6 Implementation details . . . 45
3.7 Summary and limitations . . . 47

4 Online and Real-time Action Tube Detection 49
4.1 Introduction . . . 49
4.1.1 An online, real-time framework . . . 50
4.1.2 Extension to early label predictions . . . 50
4.2 Overview of the approach . . . 52
4.2.1 Resulting contributions . . . 53
4.3 Integrated detection network . . . 53
4.3.1 Detection network design and training. . . . 53
4.3.2 Fusion of appearance and flow cues . . . 54
4.4 Online action tube generation . . . 54
4.4.1 Problem formulation . . . 54
4.4.2 A novel greedy algorithm . . . 55
4.4.3 Temporal labelling . . . 56
4.5 Optical flow computation . . . 57
4.6 Early action prediction . . . 57
4.7 Experiments . . . 57
4.7.1 Early action label prediction . . . 59
4.7.2 Spatiotemporal performance over time . . . 60
4.7.3 Global spatiotemporal performance . . . 61
4.7.4 Relative contribution of tube generation and SSD . . . 63
4.7.5 Contribution of the flow stream . . . 64
4.7.6 Test time detection speed . . . 66
4.7.7 Qualitative results . . . 67
4.8 Summary and limitations . . . 67

Part II : Micro-tubes for Action Detection and Future Prediction of Tubes **71**

5 Transition Matrix Network for Flexible Micro-tube Proposal Generation **73**

5.1 Introduction . . . 73
5.1.1 AMTNet - 3D proposals for micro-tube detection . . . 73
5.1.2 TraMNet - Flexible 3D proposals . . . 74
5.1.3 Handling sparse annotations . . . 75
5.2 Overview of the approach. . . . 76
5.2.1 Contributions . . . 77
5.3 Base network . . . 78
5.4 HMM-based action proposal generation . . . 79
5.4.1 Approximation of the HMM transition matrix . . . 80
5.4.2 Transition matrix computation for TraMNet . . . 80
5.4.3 Identity matrix for AMTNet . . . 82
5.5 Configurable pooling layer . . . 82
5.5.1 Anchor micro-tube sampling . . . 82
5.5.2 Classification and regression layers . . . 83
5.5.3 Generality at test time . . . 83
5.6 Online action tube generation . . . 83
5.6.1 Bounding box interpolation . . . 85
5.6.2 Fusion of appearance and motion cues . . . 85
5.7 Experiments . . . 85
5.7.1 Datasets and evaluation setup . . . 85
5.7.2 Network training and implementation details. . . . 86
5.7.3 Action detection performance . . . 87
5.7.4 AMTNet comparison with 2D methods . . . 88
5.7.5 TraMNet on Transformed-UCF101-24 . . . 88
5.7.6 Location invariance at test time . . . 88
5.7.7 Handling sparse annotations . . . 89
5.7.8 Training and testing at multiple Δ's . . . 90
5.7.9 Temporal labelling . . . 91
5.7.10 Detection speed . . . 91
5.8 Summary and limitations . . . 92

6 Predicting Future of Action Tubes **93**

6.1 Introduction . . . 93
6.2 Overview of the approach . . . 95

6.2.1 Resulting contributions 96
6.3 Base network 96
6.3.1 Converting AMTNet into TPNet 97
6.4 Training TPNet 97
6.4.1 Multi-task learning 98
6.5 Tube prediction framework 99
6.5.1 Problem statement 99
6.5.2 Tube prediction using TPNet 99
6.6 Experiments 101
6.6.1 Action detection performance 102
6.6.2 Early label prediction and online detection 104
6.6.3 Future action tube prediction 104
6.6.4 Test Time Detection Speed 105
6.7 Summary and limitations 106

Part III : Online/Causal 3D Representations **107**

7 Recurrent Convolution for Causal 3D CNNs **109**
7.1 Introduction 109
7.2 Overview of the approach 111
7.2.1 Resulting contributions 111
7.3 2D to 3D CNNs 112
7.3.1 Inflated 3D network (I3D) 112
7.3.2 Separated convolution networks 113
7.4 3D Recurrent convolutional network 113
7.4.1 Recurrent convolutional unit 114
7.4.2 Unrolling a recurrent convolutional network 115
7.4.3 Temporal resolution preservation 115
7.4.4 Causality and long-term dependencies 116
7.4.5 ImageNet initialisation for the 2D layers 116
7.4.6 Identity Initialisation for the Hidden Layers 116
7.5 Experiments 117
7.5.1 Implementation details 117
7.5.2 Fair training setup 117
7.5.3 Results on action recognition 119
7.5.4 Results on temporal action detection 120
7.5.5 Causality and temporal reasoning 121
7.5.6 Evolution of recurrence with network depth 123
7.5.7 Effect of weight initialisation. 124

7.5.8 Discussion 124
7.6 Summary and limitations 126

8 Conclusions and Future Research Directions 129
8.1 Summary of contributions of the thesis 129
8.1.1 Action detection using frame-level deep features 129
8.1.2 Online and real-time action detection 130
8.1.3 Flexible micro-tube proposals 130
8.1.4 Predicting Future locations of action tubes 130
8.1.5 Causal 3D representations 130
8.2 Overall conclusion 131
8.3 Future research directions 131
8.3.1 Causal spatiotemporal action detection with 3D CNNs . 131
8.3.2 Improving action representation 132
8.3.3 Improving temporal action detection 132

References 133

List of Figures

1.1 Visualisation of online spatiotemporal action detection problem in a test video. 5
1.2 Visualisation co-occurring spatiotemporal action detection problem in a test video. 5

3.1 Visualisation of action tube detections in a test video. 28
3.2 Overview of two-stream based deep action detection pipeline. . . 31
3.3 Class-specific K action paths and tubes. 35
3.4 Sample action detection results on UCF101-24. 40
3.5 Sample action detection results on JHMDB-21. 41
3.6 Action detection results on LIRIS-HARL. 42
3.7 Performance comparison between Selective Search and RPN-based region proposals. 43

4.1 Overview of online and real-time action detection framework. . . 51
4.2 Early action label prediction results. 59
4.3 Online action detection results using the AUC (%) metric on J-HMDB-21. 60
4.4 Action detection results using the mAP (%) metric. 61
4.5 Sample action detection results in "Fencing" video. 67
4.6 Sample action detection results in "Surfing" video. 68
4.7 Sample early action label prediction and online action detection results on J-HMDB-21. 69
4.8 Sample action detection results on UCF101-24. 70

5.1 Illustrating the key limitation of anchor cuboids. 74
5.2 Overview of transition matrix network (TraMNet). 77
5.3 Base network architecture for AMTNet and TraMNet. 78
5.4 Transition matrix and sampling of transition hypotheses. 81
5.5 Micro-tube interpolation and linking illustration. 84
5.6 Performance of TraMNet with various train and test Δ. 91

6.1 Future prediction of tubes, a problem statement. 94
6.2 Tube predictor network at test time. 95
6.3 Overview of the action micro-tube detection network (AMTnet). 96
6.4 Overview of the tube predictor network (TPNet) at training time. 97
6.5 Overview of future tube prediction framework. 100
6.6 Early label prediction results on J-HMDB-21. 103
6.7 Future action tube prediction results. 104

7.1 Illustration of basic build units of 3D CNNs. 110
7.2 An unrolled version of Recurrent Convolutional Network (RCN). 114
7.3 Early action label prediction results and segment-level results on Kinetics. 122
7.4 Statistics of hidden state convolution's weight matrices 123

List of Tables

3.1 Activity detection performance on validation and testing set of ActivityNet dataset. 38
3.2 Quantitative spatialtemporal action detection results (mAP). . . 39
3.3 Action detection results (mAP) on LIRIS-HARL. 42
3.4 Spatiotemporal action detection results (mAP) on UCF101-24. . 44
3.5 Test time detection speed comparison on J-HMDB-21. 45

4.1 Action detection results (mAP) on UCF101-24 dataset. 62
4.2 Action detection results (mAP) on J-HMDB-21. 63
4.3 Classwise action detection results on UCF101-24. 65
4.4 Test time detection speed. 67

5.1 Action detection results on UCF101-24. 87
5.2 Action detection results (video-mAP) on the DALY dataset. . . 89

6.1 Action detection results on JHMDB dataset. 102

7.1 Base network architecture of ResNet-18 based I3D model. 113
7.2 Action recognition accuracy on the validation set of the Kinetics with ResNet-18 as base model. 118
7.3 Action classification accuracy of different models on the validation set of the Kinetics dataset. 119
7.4 Temporal action detection performance on MultiThumos dataset. 120
7.5 Comparison of causal and anti-causal models. 121
7.6 Effect of initialisation on RCN's performance. 124

Nomenclature

Acronyms

AMTnet: Action Micro Tube Regression network.

HMS: Human Motion Segmentation.

ROI: Region of Interest.

Reg layer: Regression layer.

SS: Selective Search.

SVM: Support Vector Machine.

2PDP: Two Pass Dynamic Programming.

BoVW: Bag-of-Visual-Words.

Cls layer: Classification layer.

CNN: Convolutional Neural Network.

DPM: Deformable Part based Model.

FC (or fc) layer: Fully connected layer.

IoU: Intersection over Union.

LSTM: Long Short-Term Memory recurrent neural network.

RNN: Recurrent Neural Network.

R-CNN: Region-based CNN.

Chapter 1

Introduction

A video is a set of natural images ordered according to the time they were captured by a camera device. Videos can play the role of modern-day journals, personal journals, record books, novels, radio for sports, and letters to friends, for they can store a vast variety of information in a single format. The recording of videos is becoming increasingly simple and cheap by readily available mobile devices. There are various video libraries available these days, e.g. YouTube[1], Netflix[2]. Recordings from millions of closed-circuit cameras (CCTV) also exist. Almost 70% of the world internet traffic is video content. Because of their multi-modal nature, it is difficult to interpret videos automatically. This makes video content understanding a major problem in artificial intelligence (AI).

1.1 Understanding video content

Video understanding is a difficult task because it inherits all the complexity of the video capturing process and because of its multi-modal (video, audio, subtitles, commentary) nature. Sometime these modalities could be very useful complementary signals to visual features, e.g. audio in the professional broadcasting of sports videos could be very useful. The script of a movie may help us understand its content, along with the video. Most of the time, however, these complementary signals are either not available or confusing, e.g. music playing in a sports video. In the purview of this thesis, we will only concern ourselves with the visual content understanding of videos.

Video understanding entails multiple problems. Here, we itemise a few of the most visited problems by the research community.

- Human action/activity understanding

[1] www.youtube.com
[2] www.netflix.com

- Video retrieval
- Video compression
- Object tracking
- Optical flow computation
- Video content description
- Semantic video segmentation

The above list is by no means exhaustive, but it gives us an idea of the different tasks associated with video understanding. These problems pose multiple different types of challenges to a researcher. For instance, keeping track of moving components is necessary for object tracking, optical flow estimation and video compression. Content identification is equally necessary for video content description and action understanding.

1.2 Why human actions?

In this thesis, we focus on human action understanding in particular for two main reasons. Firstly, most of the video content is created around humans and their activities/actions, as in the case of a security guard having to flag the unusual movements of humans. Secondly, action understanding implicitly requires to concurrently solve several of the above-listed problems. Algorithms need needs to understand the content of video if a human is interacting with some objects (which involves implicit object tracking and content recognition), but also need to understand motions (which requires optical flow computation and video compression).

A variety of terms are used to describe human behaviour understanding e.g. actions, events, and activities. We think it is useful to define these terms before moving forward.

- An *event* is something that happens, not necessarily because of the will of agent (human/robot/machine); examples are: a person falling, road accident, and raining etc.
- An *action* is an event which happens because an agent (typically, a human) wanted it to happen, e.g. walking, running, throwing a ball, playing piano, working on a laptop etc.
- An *activity* is an ensemble of atomic actions performed by one or more agents, e.g. protesting on the street, playing basketball, and javelin throwing etc.

1.3 Types of problems in human action understanding

Human action understanding can be classified into various subcategories. Next, we will define some of the tasks involved in human action understanding and link them with some examples of the datasets used to study them.

1. *Action recognition* is one of the most fundamental and studied problems in human action understanding. Here, given a video clip (typically of limited duration, such as a few seconds) we need to label it with one of the categories in an action vocabulary defined by the user or the dataset. Example datasets are UCF101 [1], Sports1M [2], Kinetics [3].

2. *Temporal action segmentation* consists in labelling each frame in the video with one or more labels from a list of action categories. The task is sometimes called 'frame-wise action classification'. Example datasets include Charades [4] and MultiThumos [5].

3. *Temporal action detection* is a task in which we need to identify the temporal boundaries (start and end timestamps) of each action instance, together with its label. Given a video, we need to detect all action instances with their category labels, start time, and end time. Example datasets include, again, Charades [6] and THUMOS [7].

4. *Spatiotemporal action detection* indicates the case in which we need to identify the label, temporal boundaries and spatial bounding box in each frame associated with an action instance. Each action instance is called an *action tube* in the literature [8, 9]. An action tube is a set of bounding boxes linked across time. Some example datasets includes UCF101-24 [1], J-HMDB-21 [10], AVA [11].

5. *Spatiotemporal action segmentation* is similar to action tube detection, but also includes the pixel-level segmentation of the human involved in each action instance. J-HMDB-21 [10] is an example of a suitable dataset.

6. *Early action label prediction* is the problem of predicting the action label of a clip as early as possible, after observing a few frames from the initial part of the video. This problem can be considered as a more advanced version of the recognition problem, for here we recognise the action category by observing as few frames as possible. Most of the recognition and temporal segmentation datasets can be used to evaluate such a task.

7. *Causal/Online spatiotemporal action detection* is the online version of the spatiotemporal action detection problem, in which we detect action instances in the observed part of the video. Spatiotemporal reasoning is limited to the frames observed up to that point. Spatiotemporal action detection datasets are used to evaluate these tasks as a function of video observation percentage.

8. *Future action prediction* has many interpretations. Some of them include next action label prediction, i.e., what will be the action label a certain amount of time in future, when will the next action start, and so on. We are seeing a surge in methods designed to address these kinds of problems – we will discuss them in more detail in the next chapter, in the related work section. Recognition and detection datasets can be modified to validate these tasks.

9. *Future action tube prediction* is the problem of predicting the future bounding boxes and label of an action tube by only observing the current and past video frames. Action tube detection datasets can be employed to evaluate solutions to this problem.

In this thesis, we will touch upon most of the above problems, while mainly focusing on spatiotemporal action detection. We will start with our proposed solution to the spatiotemporal detection problem. We will then tackle the problem of solving action detection in an online and real-time fashion. We will consider multiple approaches to the online action detection problem and its extension to early label prediction and future tube prediction. Finally, towards the end of the thesis, we will present a spatiotemporal representation which is online/causal in nature.

1.4 Why online action tube detection

We argue that the problem of detecting action tubes in space and time in untrimmed videos is a rather comprehensive human action understanding problem. It encompasses problems such as action recognition, temporal action detection, the spatial detection and tracking of each actor. All these tasks require a complex understanding of the video content, as well as of its context, for context is important to understand what humans are doing: for instance, basketball and tennis are sports played in specific types of courts.

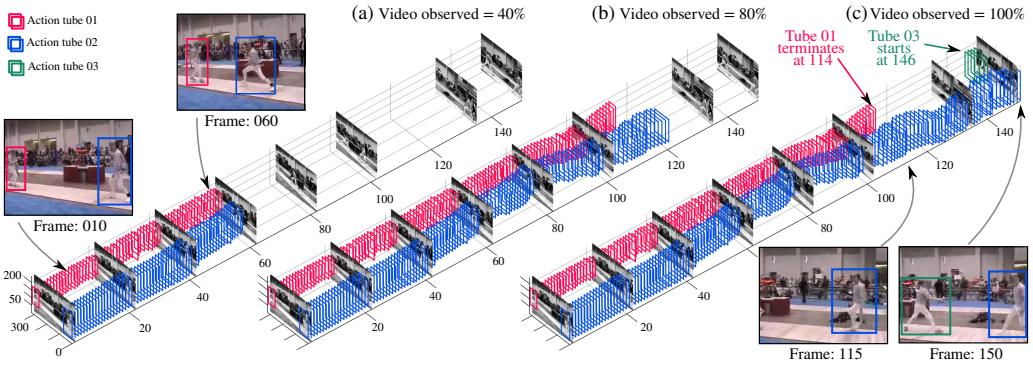


*Figure 1.1: Online spatio-temporal action detection and early action label prediction in a test 'fencing' video from UCF-101-24 [1]. **(a)** to **(c)**: A 3D volumetric view of the video showing detection boxes and selected frames. At any given time, a certain portion (%) of the entire video is observed by the system, and the detection boxes are linked up to incrementally build space-time action tubes.*

Further, we consider action-tube detection in an online mode (or 'causal' setting), where we need to process the video as it comes, based on its present and past observations, as shown in Figure 1.1. Such a style of processing is crucial for action detection system to be deployed in various real-world scenarios, including for instance action detection of other road agents for self-driving cars, human-robot interaction, surgical robotics. Moreover, if we are able to detect actions in an online fashion then it becomes relatively easier to extend the same method to the future action prediction task – we will show such an example in Chapter 6.

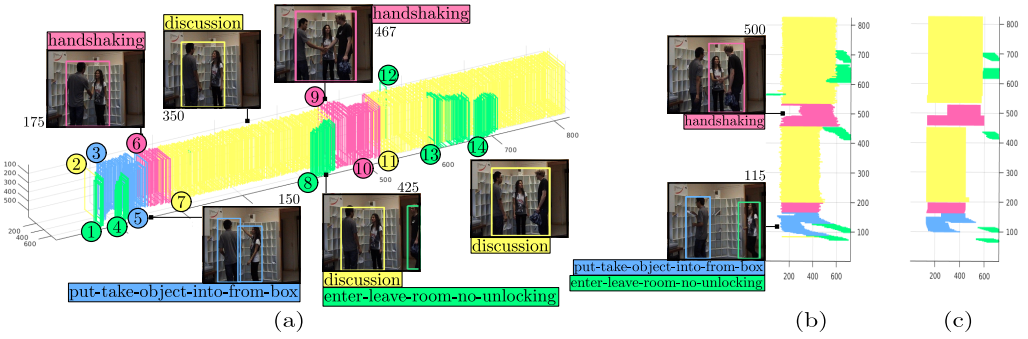


*Figure 1.2: Co-occurring spatio-temporal action detection in a test video from LIRIS-HARL [12]. **(a)** A 3D volumetric view of the video showing detection boxes and selected frames with their labels. Two people enter a room and put/take an object from a box (frame 150). They then shake hands (frame 175) and start having a discussion (frame 350). In frame 450, another person enters the room, shakes hands, and then joins the discussion. Each action tube instance is numbered and coloured according to its action category. **(b)** Action tubes drawn as viewed from above, compared to **(c)** the ground truth action tubes.*

It is important to note that the action tube detection problem involves the detection of *co-occurring* action instances. These action instances could belong

to the same action class (see, Figure 1.1) or to different action classes (see frame 115 or 425 in Figure 1.2). One class of action/interaction could turn into a different class (see instances number 6 and 7 in Figure 1.2).

We argue that an instance-based solution to the action detection problem may provide a better understanding of a video's content, and of the complex interactions and activities taking place there. Moreover, we strive to address this problem in a causal/online and real-time fashion, in order for any solution to be applicable to real-world problems.

1.5 Contributions of the thesis

In this thesis, we describe the first online and real-time action detection system, able to detect multiple co-occurring actions in untrimmed videos, unlike previous works [8,13] which were offline in nature. We establish that supervised proposals are important for action detection [14], and that temporal localisation can be formulated as an efficient dynamic programming-based solution, in opposition to earlier sliding window approaches. Next, we show how we can extend the approach to action detection based on individually processing each video frame to methods that can process multiple frames at the same time. In particular, we introduce a new way of generating cross-frame flexible anchor proposals, and compare it against the cuboid anchor proposals in [14–16]. Also, for the first time, we introduce the future action tube prediction problem as an extension of online action tube detection problem. Finally, we introduce an online 3D spatiotemporal video representation based on a causal 3D CNN architecture which, while being causal, is competitive with the previous a-causal 3D representations [17–19].

In this thesis, each contribution is separately presented in each chapter in the context of the previous related work. Each chapter has a subsection called "resulting contributions" in the "overview of the approach" section, where the contributions of each chapter are spelled out in detail. We also run a comparison with the relevant previous state of the art in a "related work" chapter (Chapter 2). Finally, we summarise these contributions again in Chapter 8, in which the contributions of each chapter are consolidated in Section 8.1.

1.6 Overview of the thesis

First and foremost, all related work is reviewed in chapter (Chapter 2). The five main contribution chapters of this thesis follow. These chapters are grouped into three parts, Part I, Part II and Part III. Finally, our contributions and possible future extensions are summarised in Chapter 8.

Part I 2.10.4 includes two chapters (Chapter 3, and Chapter 4). The main theme of Part I is online and real-time action detection. Chapter 3 describes an offline action detection approach based on previous works [14, 20]. Here, we lay the groundwork for the next chapters, and a possible extension towards an online and real-time approach. We show that supervised proposals are important for action detection [14], and temporal localisation can be efficiently formulated in a dynamic programming setting, in contrast with the earlier sliding window approaches.
In Chapter 4 we present what was, at the time of publication, the first online and real-time action detection system, based on our published work [9]. We bring forward a host of changes to make the action detection system online and real-time, while being extremely competitive with other, offline systems in terms of performance. In particular, we: (i) propose an online tube generation algorithm; (ii) exploit the real-time network architecture of single-shot-detector (SSD) [21]; (iii) ablate the replacement of expensive optical flow calculation approaches [22] with real-time optical flow method [23].

Part II 4.8 also includes two chapters (Chapter 5, and Chapter 6). Its main theme is the notion of action 'micro-tube' for action detection and future prediction. Chapter 5 presents two approaches for extending single frame-based action detectors to multi-frame action detectors, based on two other published works of ours [24, 25]. The first approach (based on [14]) involves generalising a frame-based object/action detector to a multi-frame detector by extending conventional frame-level anchor proposals in the temporal direction to generate anchor-cuboids. The main idea is that predictions made in reference to an anchor cuboid are considered linked (to form micro-tubes, as pair of bounding box detections with an attached label). This provides us with pairs of boxes which are inherently linked across frames, a step towards doing away with the temporal linking in a post-processing stage. We will call such a detector an *action micro-tube network* (AMTNet). The next approach introduces flexible anchor proposals across frames, which allow the anchor in one frame to shift to another location in the next frame, in order to handle "dynamic" actions. This is done by learning a transition probability matrix from the training data in a hidden Markov model formulation, leading to an original configurable layer architec-

ture. We call such an action detector a *transition matrix network* (TraMNet) [25].
Building on these results, Chapter 6 shows that a micro-tube prediction network can be extended to predict the future of each action tube individually. We formulate this as a multi-tasking problem. A new task head is added to the micro-tube prediction architecture to predict the future locations of each micro-tube. The resulting network is called a *tube predictor network* (TPNet) [26]. Further, we propose a tube prediction framework which completes the future of each tube by taking the outputs of the individual micro-tubes. We show that TPNet can cope with the intrinsic uncertainty about the future better than the considered baseline methods, while remaining state-of-the-art in the action detection task.

Lastly, we have a single chapter (Chapter 7) in Part III, where we take steps to transform state-of-the-art three dimensional (3D) convolutional neural networks (CNNs) to causal 3D CNNs, in line with the main objective of this PhD work, which is online action detection. The fundamental observation is that 3D CNNs are anti-causal (i.e., they exploit information from both the past and the future to produce feature representations, thus preventing their use in online settings). They also constrain the temporal reasoning horizon to the size of the temporal convolution kernel, and are not temporal resolution-preserving for video sequence-to-sequence modelling, as, e.g., in action detection. To address these serious limitations, in Chapter 7 we present a new architecture for the causal/online spatiotemporal representation of videos. Namely, we propose a novel *recurrent convolutional network* (RCN) [27] which relies on recurrence to capture the temporal context across frames at every level of network depth. We show in our experiments on the large-scale large "Kinetics" and "MultiThumos" datasets that the proposed method achieves superior performance even when compared with anti-causal 3D CNNs while being causal and using fewer parameters.

1.7 Structure of the chapters

All the main contribution chapters (Chapter 3 to Chapter 7) follows the same overall structure. Firstly, we have an 'introduction' section, where we present motivation in terms of applications and previous works. It also contains an 'outline' of that particular chapter. Next, we have 'overview of the approach' section, where we highlight the main steps included in our approach. This section also includes a subsection for 'resulting contributions' from our approach. Following that, various sections describe the approach. An experiment section

follows in which we evaluate and ablate our approach in the context of the claims made in the approach and introduction sections. Finally, each chapter has a 'summary and limitations' section, which includes a short summary of the chapter, the limitations of the presented approach, and a paragraph where we look ahead to the next chapter in anticipation.

1.8 Resulting publications and resources

1.8.1 List of publications

Part of thesis: Following is the list of publications that make the major part of this PhD thesis.

1. **Gurkirt Singh** and Fabio Cuzzolin, "Recurrent Convolutions for Causal 3D CNNs", in Proceedings of International Conference on Computer Vision Workshop (ICCVW) on Large Scale Holistic Video Understanding, 2019.

2. **Gurkirt Singh**, Suman Saha and Fabio Cuzzolin, "TraMNet - Transition Matrix Network for Efficient Action Tube Proposals", in Proceedings of Asian Conference on Computer Vision (ACCV), 2018.

3. **Gurkirt Singh**, Suman Saha and Fabio Cuzzolin, "Predicting Action Tubes", in Proceedings of European Conference on Computer Vision Workshops (ECCVW) on Anticipating Human Behaviour Workshop, 2018.

4. **Gurkirt Singh**, Suman Saha, and Fabio Cuzzolin, "Online Real-time Multiple Spatio-temporal Action Localisation and Prediction", in proceedings of International Conference on Computer Vision (ICCV) 2017.

5. Suman Saha, **Gurkirt Singh** and Fabio Cuzzolin, "AMTnet: Action-Micro-Tube Regression by end-to-end Trainable Deep Architecture", in proceedings of International Conference on Computer Vision (ICCV) 2017.

6. Suman Saha, **Gurkirt Singh**, Michael Sapienza, Philip Torr and Fabio Cuzzolin, Deep Learning for Detecting Multiple Space-time Action Tubes in Videos, proceedings of British Machine Vision Conference (BMVC) 2016.

7. **Gurkirt Singh** and Fabio Cuzzolin, "Untrimmed Video Classification for Activity Detection: Submission to ActivityNet Challenge", in Tech-report arXiv:1607.01979, presented at ActivityNet Challenge Workshop in CVPR 2016.

Other works: Below is the list of other works which were conducted during PhD but these are not part of this thesis.

1. Harkirat Behl, Michael Sapienza **Gurkirt Singh**, Suman Saha, Fabio Cuzzolin and Philip Torr, "Incremental Tube Construction for Human Action Detection", British Machine Vision Conference (BMVC), 2018.

2. **Gurkirt Singh***, Stephen Akrigg*, Valentina Fontana*, Manuele Di Maio, Suman Saha, Fabio Cuzzolin,"Action Detection from a Robot-Car Perspective", Preprint arXiv: 1807.11332, 2018.

3. Silvio Olivastri, **Gurkirt Singh** and Fabio Cuzzolin,"An End-to-End Baseline for Video Captioning", in Proceedings of International Conference on Computer Vision Workshop (ICCVW) on Large Scale Holistic Video Understanding, 2019.

4. Suman Saha, **Gurkirt Singh**, Michael Sapienza, Philip Torr and Fabio Cuzzolin,"Spatio-temporal Human Action Localisation and Instance Segmentation in Temporally Untrimmed Videos", arXiv preprint arXiv:1707.07213, 2017.

1.8.2 List of software packages and other resources

The following software packages from this thesis are available online.

- Source code for our ICCV 2017 [9] work is publicly available online at: `https://github.com/gurkirt/realtime-action-detection`.
  CNN training and evaluation code are developed using Pytorch and Python.
  The tube construction algorithm is available in both Matlab and Python.

- ICCV 2017 work [9] - YouTube demo video link `https://www.youtube.com/watch?v=e6r_39ETe-g`.

- Annotation of UCF101-24 [1] were corrected during for our ICCV 2017 [9] work. The corrected annotations at available at `https://github.com/gurkirt/corrected-UCF101-Annots`.

- Source code for our ActivityNet challenge submission during CVPR 2016 [20] work is publicly available online at `https://github.com/gurkirt/actNet-inAct`.
  The source code is developed in Python.

- Source code for our BMVC 2016 [14] work is publicly available online at: `https://bitbucket.org/sahasuman/bmvc2016_code`.
  Source code developed using MatCaffe (the Matlab wrapper for Caffe deep learning toolbox).

- BMVC 2016 work [14] - YouTube demo video link `https://youtu.be/vBZsTgjhWaQ`.

Chapter 2

Related Work

In this chapter, we present a literature review from various perspectives concerning the problems and techniques mentioned in the Introduction. We review the work most related to the concepts and contributions of this thesis, as described in the remainder of the thesis.
The terms "previous work" and "recent work" are used in the context of the work presented in this thesis – for instance, the paper [28] on action detection was published before our relevant publication [20], and is therefore considered 'previous' work. However, [29] appeared after [20], therefore it is consider 'recent' work.

We first briefly review the most prominent work in action recognition in Section 2.1.1, as the most basic video clip understanding problem. We then outline the recent advances in temporal action detection. Finally, we review the state of the art in spatiotemporal action detection.

2.1 Action recognition

Action recognition, also called *video classification*, is most widely studied of all video action understanding problems (§ 1.3).

2.1.1 Traditional approaches

Action recognition has been studied for a long time [30–32]. Seminal work by I. Laptev [32] provided a solid platform for subsequent important papers [33, 34] in action understanding. As a result, image-based feature descriptors (such as HOG [35] and SIFT [36])) were adopted or suitably extended to process videos, e.g. as in motion boundary histograms (MBH) [37]. 3D descriptors [38] and dense trajectory features *et al.* [39, 40] were later proposed, which would exploit Bag-of-Visual-Words (BoV) [41] or Fisher vector (FV) [42] representations.

2.1.2 2D CNNs

After the advent of deep learning [43–45], image-based convolutional neural networks (2D-CNN) were adopted to tackle the action recognition problem. Simonyan and Zisserman [46] proposed to train two separate 2D-CNNs on RGB and optical flow images as inputs: their approach was able to produce comparable to local feature-based approaches [40]. Feichtenhofer *et al.* [47] showed that fusing the two streams can improve the performance further. Efforts were also made to better capture temporal information with 2D CNNs. For instance, Donahue used LSTMs [48] on top of 2D CNN features. Wang *et al.* [49] proposed instead to train 2D CNNs with segment-level inputs. Other approaches include, among others, CNN features in combination with LSTMs [50] for temporal action detection, 2D features used in an encoder-decoder setup along with temporal convolutions [29], and conditional random fields on series of 2D features [51] for temporal action detection and recognition. All these methods showed promising results. In all these architectures, however, the optical flow stream and the few layers on the top of the 2D features are the only sources of temporal reasoning.

2.1.3 3D CNNs

Initial attempts to apply three dimensional (3D) CNNs models [52, 53], which promised to be able to perform spatial and temporal reasoning in parallel, met limited success. Later, Carreira *et al.* [17] improved the existing 3D CNNs by emplying ImageNet-based initialisation and by training their models on the large-scale Kinetics dataset [3]. The resulting models outperformed older 2D ones. In spite of this, 3D CNNs remain heavy and very expensive to train – e.g., 64 GPUs were used in [17].

In alternative, the notion of factorising 3D convolutional networks was explored by Sun *et al.* [54]. This inspired [18,19,55] to decompose 3D convolutions into 2D (spatial) and 1D (temporal) convolutions. Recent work by Xie *et al.* [18] has further promised to reduce complexity (in terms of number of parameters) while making up for the performance lost via a gating mechanism. Tran *et al.* [19] would keep the number of parameters equal to that of 3D convolutions, but boost performance by increasing the number of kernels in the 2D layer. The size of the temporal convolution kernel needs to be fixed to a relatively small number (e.g., 3 in [17–19]). Varol *et al.* [56] have thus proposed the use of long-term convolutions to capture long-range dependencies in the data. Wang *et al.* [57], instead, have introduced non-local blocks in existing 3D CNN architectures, to capture the non-local context in both the spatial and

the temporal (present, past and future) dimensions. Feichtenhofer *et al.* [58] have proposed to combine the information coming from two branches of 3D CNNs which operate at different frame rates to boost the performance of action recognition even further. Similarly, Diba *et al.* [59] have suggested to combine information at multiple stages of a 2D CNN with that of a 3D CNN.

2.2 Temporal action detection

In the past, the temporal action detection problem was mostly tackled using expensive sliding window approaches [33, 60–63]. These can deliver good results [60–62], but are too inefficient to work in real-time. Recently, deep learning-based methods have led to significant advances in this area as well. For instance, Shou *et al.* [64] have employed 3D CNNs [52, 53] to address temporal action detection in long videos. Recurrent neural networks (RNN) [65] and LSTMs [66] are also increasingly being used [67–70] to address the problem, as well as temporal convolutions [71]. In past, dynamic programming has been employed to solve the problem efficiently [20, 72, 73]. Some of the above works [5, 67, 73] can perform action detection in an online fashion. In our work, specifically in Chapter 3, we adopt [73] for the temporal detection of action tubes.

More recently, temporal proposal-based methods [74–80] have gained much traction and have proven to be very effective [76, 79, 80]. The philosophy of Faster R-CNN [81] has been extended to videos to predict the start and end time of action instances in untrimmed videos by [77, 79]. Zhao *et al.* [76] would generate proposal by applying a watershed algorithm [82] on the frame-level actionness (presence of action) of video frames. Similarly, Lin *et al.* [80] would generate proposals by predicting a score for the likelihood of each frame to belong to the start or the end of an action instance.

Currently, proposal-based methods [76, 80] are state-of-the-art in temporal action detection.

2.3 Multi-label temporal action segmentation

Although relatively new, the task of dense temporal prediction has been studied by several authors [5, 29, 51, 64, 83] in a multi-label setting, i.e., in which each video may contain multiple action labels. Two major datasets related to

this task are Charades [4] and MultiThumos [5]. Yeung *et al.* [5] has recently proposed to this purpose a multiple-output version of LSTMs, named multi-LTSM [5]. The authors of [29, 84] use temporal deconvolution layers on top of the C3D network [53] to recover the loss of temporal resolution due to the temporal convolutions in the base C3D network [53]. Piergiovanni *et al.* [83] have proposed an attention mechanism in the form of temporal structure filters that enable the model to focus on particular sub-intervals of a video. However, their model requires the pre-computation of 3D CNN features for the entire video.

In this thesis we show an online method able to solve the above problem more efficiently in an online/causal/streaming manner in Chapter 7.

2.4 Spatiotemporal action detection

Spatiotemporal action detection is also a widely studied problem. Later in this thesis we provide a more extensive overview of the methods related to it. Here, we first quickly review frame-level action detection methods in Section 2.4.1, to later move on to multi-frame methods in Section 2.4.2. Lastly, we consider methods based on 3D spatiotemporal representations in Section 2.4.3.

2.4.1 Single-frame based methods

Initial proposals for human action detection included human location detection-based approaches such as [61, 85–88]. There, after a human was detected in the video, 3D features [38] and dense trajectories *et al.* [39] were pooled into a feature vector representation [41, 42]. Weinzaepfel *et al.*'s work [13] performed both temporal and spatial detections by coupling frame-level EdgeBoxes [89] region proposals with a tracking-by-detection framework. In their work, however, temporal trimming was still achieved via a multi-scale sliding window over each track, making the approach inefficient for longer video sequences.
More recently, Saha *et al.* [14] and Peng *et al.* [90] made use of supervised region proposal networks (RPNs) [81] to generate region proposals for actions at the frame level, and solved the space-time association problem via 2 recursive passes over frame-level detections for the entire video by dynamic programming. The use of a non-real-time and 2-pass tube generation approach, however, made their methods intrinsically offline and inefficient.

In this thesis, we extend our previous offline frame-level work [14] to an online and real-time action detection method [9] in Chapter 4. Our framework employs

a real-time optical flow (OF) algorithm [23] and a single shot SSD detector [21] to build multiple action tubes in a fully incremental way, and in real-time, unlike previous methods. After [9] was published, several efforts [15, 16, 24, 91] were directed towards exploiting the spatiotemporal information encoded by multiple frames. The reason is that, in frame-based methods, the deep network is not in a condition to learn the temporal features essential for accurate action detection, and temporal reasoning is performed by some tube construction stage, in a sub-optimal post-processing step.

2.4.2 Multi-frame based methods

Most early attempts to solve space-time action detection using information coming from multiple frames were based on action cuboid hypotheses and sliding-window based approaches [33, 61, 92–94]. Their main limitation was the assumption that an action can be localised using a cuboidal video sub-volume, together with the fact that a sliding window-based approach is very expensive.

Very recently, video-based action representation approaches have appeared [15, 16, 24, 91] which address this issue by learning complex non-linear functions (in the form of CNNs) which map video data (instead of image data) to a high dimensional latent feature space. Kalogeiton *et al.* [15] and Hou *et al.* [16]'s models map K video frames to such a latent space, whereas Saha *et al.* [24] only require 2 successive frames to learn spatio-temporal feature embeddings. The main advantage of [24] over [15, 16] is that its framework is not constrained to process a fixed set of K frames, but is flexible enough to select training frame pairs $(f_t, f_{t+\Delta})$ at various intervals by varying the Δ value according to the requirements of a particular dataset (i.e., a larger Δ for longer sequences, or a relatively smaller one for shorter video clips). As in older approaches [33, 61, 92], these more recent methods [15, 16, 24, 91] assume that 3D cuboidal anchors can be used to localise action instances in K frames.

In Chapter 5, we show that such an assumption is unrealistic for "dynamic" actions. We then present a solution able to generate flexible anchor proposals to counter this problem, based on our work in [25]. We show that cuboidal anchor-based methods are just a special case of our more general solution [25].

More recently, Lin [95] have proposed a two-stage approach for action detection. Firstly, action tubes are linked using frame-level detections; then, tube classification is performed using a recurrent network. The resulting performance improvement, however, is marginal, and even lower when compared with the base same network (VGG [96]) used in other works [15, 25].

2.4.3 3D representations for action detection

In Section 2.1.3, we learned that 3D CNNs are state-of-the-art in action recognition. In fact, spatiotemporal 3D representations [17–19, 57, 58, 97] have recently emerged as a dominant force in action detection as well [11, 98, 99]. Gu *et al.* [11], for instance, combine an inflated 3D (I3D) network [17] with 2D proposals as in [90] to exploit the representational power of I3D. A similar notion is proposed in [18,95,98]. Duarte *et al.* [99], on their part, have proposed an interesting 3D video-capsule network for frame-level spatiotemporal action segmentation. Wu *et al.* [97] combine a 3D representation with temporal filter banks to improve action detection performance. It is important to note that the above-mentioned methods are not specifically designed for action detection – most of them (except [97–99]) exhibit superior performance due to the high representational power that comes with a 3D representation.

2.5 Online and real-time methods

Relatively few efforts have been directed at simultaneous real-time action detection and classification. Zhang *et al.* [100], for example, would accelerate the two-stream CNN architecture of [46], performing action classification at 400 frames per second. Yu *et al.* [101] evaluated their real-time continuous action classification approach on the relatively simpler KTH [102] and UT-interaction [103] datasets. The above methods, however, were not able to perform spatiotemporal action detection.

To the best of our knowledge, the method we propose in Chapter 4 is the first to address the spatiotemporal action detection problem in real time.

2.6 Early action prediction and detection

Prior to 2016 early action prediction was studied by a relatively small number of authors [50, 104–107]. Ryoo *et al.* [104] used a dynamic bag of words approach to speed up their action prediction pipeline based on handcrafted features. Hoai [105] proposed their max-Margin early event detectors based on structured output SVM. In [106], Lan *et al.*proposed a way to capture human movement before the action actually happens, at a different level of temporal granularity, and were able to make predictions by observing the action for 50% of its duration. Different types of loss function were proposed to encourage

LSTMs [66] to predict the correct label as early possible in [50, 107]. None of these approaches, however, perform online spatial-temporal action detection.

More recently, Soomro *et al.* [108] have proposed an online method which can predict an action's label and location by observing a relatively smaller portion of the entire video sequence. However, [108]'s method works only on temporally trimmed videos and not in real-time, due to the expensive segmentation stage required. Singh *et al.* [9], on the other hand, have brought forward an online action detection and early label prediction approach which works on temporally untrimmed videos. Similarly, Behl *et al.* [109] solve online detection by formulating it as a multi-target tracking problem. Subsequent works [15, 25, 26] have adopted Singh *et al.* [9]'s online tube generation algorithm for online action detection. As mentioned, this approach [9] is presented in Chapter 4.

2.7 Future prediction problems

The computer vision community is in fact witnessing rising interest in problems such as future action label prediction [5, 69, 104–107, 110–113], online temporal action detection [5, 50, 67, 114], online spatio-temporal action detection [9, 108, 115], future representation prediction [111, 116] or trajectory prediction [117–119]. Although all these problems are interesting, and definitely encompass a broad scope of applications, they do not entirely capture the complexity involved by many critical scenarios including, e.g., surgical robotics or autonomous driving. In opposition to [9, 108] which can only perform early label prediction and online action detection, we will combine early label prediction, online action detection, and trajectory prediction into one problem in Chapter 6.

2.8 Causal representations

The use of temporal convolutions in all the above methods is, however, inherently anti-causal [120], as it requires information from both past and future. Any online or future prediction method needs to be causal in nature. All 2D representations-based approaches [9, 46, 47, 68, 121] are causal as they do not require future frames to make a prediction about the present. However, as mentioned, the optical stream and the few layers on the top of the 2D features are the only sources of temporal reasoning. Which is why using a 3D representation is highly desirable to improve performance – unfortunately, all existing such representations are not causal. Relevantly, Carreira *et al.* [120] have recently

proposed to address the anti-causal nature of 3D CNNs by predicting the future and utilising the flow of information in the network. They train their causal network to mimic a 3D network – the resulting performance drop, however, is significant.

In Chapter 7 we propose a Recurrent Convolutional Network (RCN) designed to solve exactly the above problem.

2.9 Related datasets

Action detection and recogntion research is evaluated with various different datasets. Here, we describe the properties of the datasets used in this thesis.

2.9.1 UCF101-24

UCF101-24 is a subset of UCF101 [1], one of the largest and most diversified and challenging action datasets. A subset of 24 classes out of 101 comes with spatiotemporal detection annotation, released as bounding box annotations of humans for the THUMOS-2013 challenge[1]. Although each video only contains a single action category, it may contain multiple action instances (up to 12 in a video) of the same action class, with different spatial and temporal boundaries. On average there are 1.5 action instances per video, each action instance covering 70% of the duration of the video. For some classes, instances average duration can be as low as 30%. We will use this dataset to evaluate the spatiotemporal action detection problem, similar to previous works [13, 122], we test our method on split 1.

Note that, while conducting our tests, we noted and corrected some issues we found with the original annotation of UCF101-24 [1] released for the THUMOS-2013 [123] challenge. The corrected annotation is available on Github[2]. These annotations were realised together with our ICCV 2017 paper [9]. UCF101-24 dataset is widely used in this thesis to evaluate spatiotemporal action detection performance.

2.9.2 ActivityNet

ActivityNet [6] dataset is a large scale temporal activity detection dataset with 200 activity classes and 100 videos per class, on average. In practice, a large proportion of videos have a duration between 5 and 10 minutes [6]. The dataset is divided into three disjoint subsets: training, validation and testing, following

[1] http://crcv.ucf.edu/ICCV13-Action-Workshop/download.html
[2] https://github.com/gurkirt/corrected-UCF101-Annots

a ratio of 2:1:1. The authors of the dataset organised a challenge in 2016 and we participated with our temporal detection formulation, securing second place. We used ActivityNet dataset to evaluate temporal action detection performance in Chapter 3.

2.9.3 J-HMDB-21

J-HMDB-21 [10] is a subset of the HMDB-51 dataset [124] with 21 action categories and 928 videos, each containing a single instance of action tube, and each video trimmed to the action's duration. Hence, it does not require temporal trimming step (§ 3.5.2). As in [8] we pick the top 3 action paths from each class to be evaluated against the ground truth tube in each video. We use J-HMDB-21 dataset in this thesis to evaluate trimmed spatiotemporal action detection performance in various chapters.

2.9.4 LIRIS-HARL

LIRIS-HARL is a human activity detection dataset [12] with 107 training and 58 testing video sequences. The videos in LIRIS HARL have a relatively longer duration than the videos in UCF101-24 and J-HMDB-21 dataset. Videos contain multiple concurrent action instances from different action classes. The average duration of action instances is much shorter than the duration of the videos: hence, this dataset is well suited to evaluate the temporal trimming capabilities of our method. We use LIRIS-HARL dataset in Chapter 3 in this thesis to evaluate the spatiotemporal action detection performance of multiple different co-occurring actions.

2.9.5 DALY

The **DALY** dataset was released by Weinzaepfel *et al.* [125] for 10 daily activities and contains 520 videos (200 for test and the rest for training) with 3.3 million frames. Videos in DALY are much longer, and the action duration to video duration ratio is only 4%, compared to UCF101-24's 70%, making the temporal labelling of action tubes very challenging. The most interesting aspect of this dataset is that it is not densely annotated, as at max 5 frames are annotated per action instance, and 12% of the action instances only have one annotated frame. As a result, annotated frames are 2.2 seconds apart on average ($\Delta = 59$). We use this dataset to evaluate our sparse annotation handling models in Chapter 5.

2.9.6 Kinetics

The Kinetics dataset comprises 400 classes and $260K$ videos; each video contains a single atomic action. Kinetics has become a *de facto* benchmark for recent action recognition works [17–19,57,120]. The average duration of a video clip in Kinetics is 10 seconds. It is used for the action classification task, we will use this dataset in Chapter 7 to evaluate our causal 3D CNN model.

2.9.7 MultiThumos

The MultiThumos [5] dataset is a multilabel extension of THUMOS [126]. It features 65 classes and 400 videos, with a total duration of 30 hours. On average, it provides 1.5 labels per frame, 10.5 action classes per video. Videos are densely labelled, as opposed to those in THUMOS [126] or ActivityNet [28]. MultiThumos allows us to show the dense prediction capabilities of RCN on long, real-world videos. We use this dataset in Chapter 7 to evaluate our causal 3D CNN model for the task of action segmentation.

2.10 Evaluation metrics

In this thesis, we use standard evaluation metrics following the previous works or variation of these metric. Next, we describe the evaluation metrics used in this thesis.

2.10.1 Classification accuracy

The classification accuracy is used to evaluate tasks of action recognition or prediction. Given an observation video clip or frames, we predict one class label with the highest score. The classification accuracy measures what percentage of our predictions are correct of all the test videos. We use classification accuracy as a metric for Kinetics dataset in Chapter 7. Also, we use it in other chapters for UCF101-24 dataset for the tasks of action recognition, early label prediction, and future label prediction.

2.10.2 Mean average precision (mAP)

Similar to other detection tasks (object detection and human detection), we use mean average-precision(mAP) as the main evaluation metric for both temporal or spatiotemporal detection tasks. Average precision is the average of the maximum precision at different recall values. The recall for a ground truth instance

is measured against an overlap threshold. In our case, the overlap is measured as the intersection-over-union (IoU) between ground truth instances and detections produced by the detection framework. Temporal-intersection-over-union (T-IoU) in temporal detection is defined over the temporal extent. In the case of spatial detection, instead, we measure the spatial IoU between ground truth bounding boxes and detected bounding boxes in a detected instance over the temporal duration of an action instance. We then take the average of the spatial IoUs to obtain an averaged spatial-IoU (aS-IoU) measure. Finally, we multiply S-IoU and TIoU to get a measure of spatiotemporal-intersection-over-union (ST-IoU). If a detected tube has the same label as the ground truth, and ST-IoU is greater than the chosen threshold (δ), then it is classified as true-positive; otherwise, it is considered a false positive. All the true positives and false positives are then sorted in decreasing ordered based on the score of each detection to compute the average precision for each class. The difficulty of detection increases with any increase in the detection threshold (δ) – a value of 0.5 (i.e., a 50% overlap with ground truth) is standard.

2.10.3 Completion-mAP

Completion-mAP is similar to mAP, but here tubes are generated by only observed part of the video. The entire tube predicted (by observing only a small portion (%) of the video) is compared against the ground truth tube for the whole video. Based on the detection threshold we can compute mean-average-precision for the complete tubes, we call this metric *completion-mAP* (c-mAP). We will use this metric in Chapter 6.

2.10.4 Prediction-mAP

Similar to c-mAP, prediction-mAP is used to evaluate the prediction made about the future of the action tubes. We measure how well the future predicted part of the tube localises. In this measure, we compare the predicted tube with the corresponding ground truth future tube segment. Given the ground truth and the predicted future tubes, we can compute the mean-average precision for the predicted tubes, we call this metric *prediction-mAP* (p-mAP). We will use this metric in Chapter 6.

Part I : Towards Online Action Detection

Chapter 3

Deep learning for Spatiotemporal Detection of Multiple Action Tubes

3.1 Introduction

We maintained in Chapter 1 that 'action tube' [8] detection provides a holistic approach to the human action understanding problem. We defined an action tube to be a set of connected bounding boxes, covering the temporal extent of an action instance. A video can contain multiple instances of the same action, as well as instances of different actions. A pictorial representation of the action tube detection problem is shown in Figure 3.1. In this chapter, we propose to tackle this problem with the help of what were in 2016 the latest CNN architectures. The following chapters will build upon these ideas to achieve better, faster, and more elegant solutions to action tube detection and future prediction of action tubes.

Advances in object detection via CNNs which took place in 2014-16 [127] have triggered a significant performance improvement in the state-of-the-art action detectors [8,13]. However, these approaches had two significant drawbacks. First, their accuracy was limited by their reliance on unsupervised region proposal algorithms such as Selective Search [8] or EdgeBoxes [13] which, besides being resource-demanding, cannot be trained for a specific detection task and are disconnected from the overall detection objective. Second, temporal detection was either achieved by expensive sliding window approach in [13] or is not taken care as in [8].

To overcome these issues we propose a novel action detection framework which, instead of adopting an expensive multi-stage detection frameworks [127],

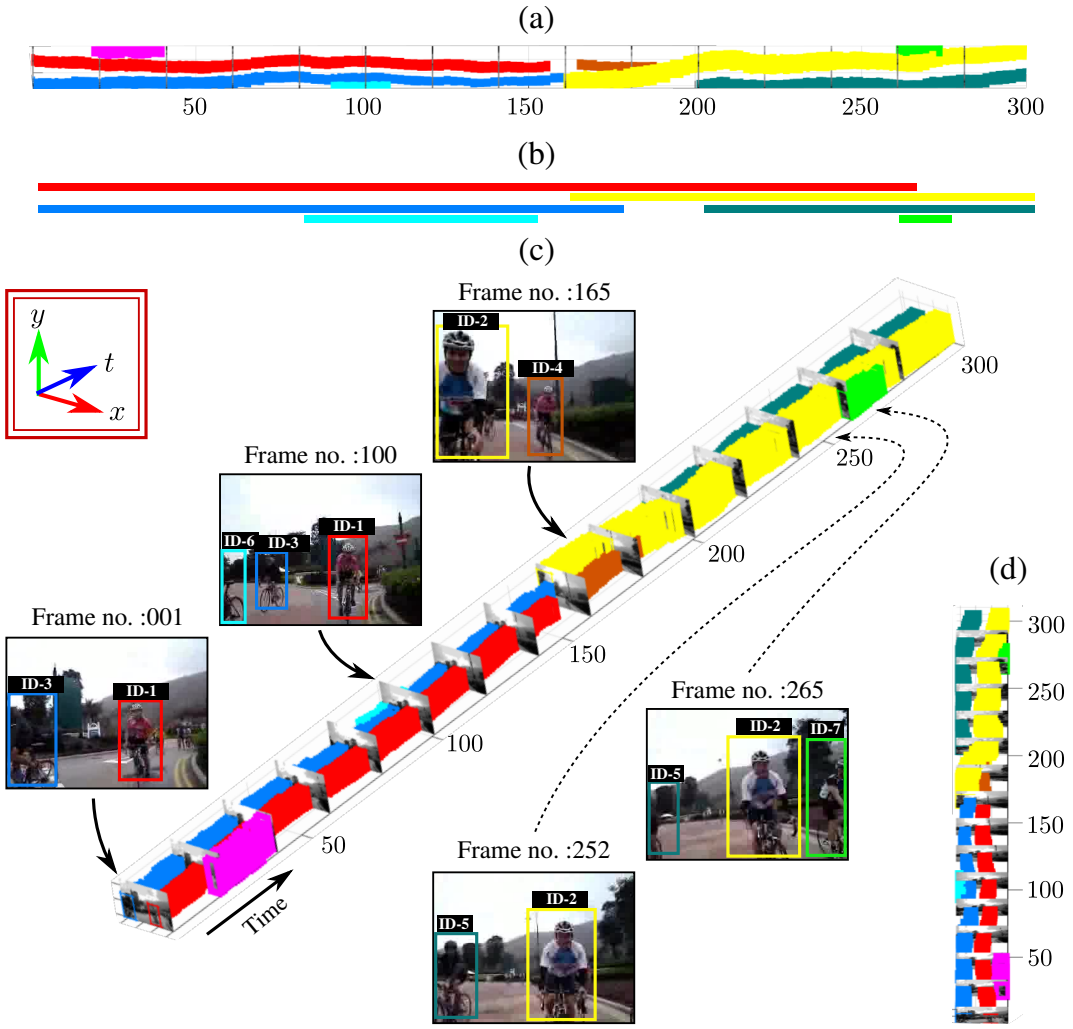


*Figure 3.1: Visualisation of action tube detections in a test 'biking' video taken from UCF101-24 [1]. **(a)** Side view of the detected action tubes, in which each colour represents a particular instance. The detection boxes in each frame are linked up to form space-time action tubes. **(b)** Illustration of the ground truth temporal duration for comparison. **(c)** Viewing the video as a 3D volume with selected image frames; **(d)** Top-down view.*

takes advantage of the most recent end-to-end trainable architectures for object detection [81], in which a single CNN is trained for both detecting and classifying frame-level region proposals in an end-to-end fashion. Detected frame-level proposals are subsequently linked and trimmed to form space-time action tubes by solving two optimisation problems via dynamic programming.

3.1.1 Action tube detection approach

We propose to solve the action tube detection problem by dividing it into three sub-problems. Firstly, we employ supervised proposal generation network to detect the action bounding boxes within each video frame. Secondly, we connect the detected bounding boxes within a suitable optimisation framework [8] to

form video-long action paths, using dynamic programming. Lastly, we trim the resulting action paths to the temporal extent of action instances via another dynamic programming-based solution [73].

We demonstrate that the proposed action detection pipeline is at least 5× faster in test time detection speeds as compared to previous works [8,13], and it can be trained efficiently as well for the same reason. In Section 3.6.5, we compare the computing time requirements during testing of our approach with [8,13] on the J-HMDB-21 [10] dataset. Moreover, our pipeline consistently outperforms [1] previous state-of-the-art results (Section 3.6).

3.1.2 Temporal action detection approach

Inspired by [72], Evangelidis *et al.* [73] proposed a simple label consistency constraint over frame-level gesture class scores, and solved the optimisation with dynamic programming in multiclass setting. We propose to use a similar formulation for temporal action detection. The difference, however, is that we tackle temporal consistency for k classes independently. This class-wise temporal detection method is used to find the temporal bounds (start and end time) of each action tube – label consistency for all constituting detection boxes is ensured. In contrast with other approaches [13,33,61] which achieve temporal trimming via an inefficient, multi-scale sliding window, we rely on a simple and efficient dynamic programming formulation.

We demonstrate that our temporal trimming solution provides performance improvement in spatiotemporal action detection on the LIRIS-HARL and UCF101-24 datasets, as well as on the temporal activity detection task part of the 2016 Activitynet [28] challenge [2].

Related publications: The work presented in this chapter appeared in BMVC 2016 [14] and [20] as part of our Activitynet [28] challenge 2016[3] submission. Our entry secured second place in the activity detection task at a detection threshold of 0.5, and first place when detection performance was averaged across multiple detection thresholds, from 0.5 to 0.95. This dissertation's author was the primary investigator in [20] and co-author in [14], to which he contributed the dynamic programming-based solution [20] to the temporal trimming of the action tubes, explained in Section 3.5.

[1] The model presented in this chapter appeared in BMVC 2016 and it outperformed the state-of-the-art results at that time.

[2] We compare ourselves with the methods published before Activitynet 2016 `http://activity-net.org/challenges/2016/data/anet_challenge_summary.pdf`

[3] http://activity-net.org/challenges/2016/

Outline: the rest of the chapter is organised as follows. We start by presenting an overview of our action tube detection approach in Section 3.2. Next, we describe the three stages of our action tube detection pipeline. Stage one (§ 3.3) includes an end-to-end trainable frame-level detector used to produce bounding box detections around humans. Stage two (§ 3.4) is the process of connecting detection bounding boxes into video long *action paths*. Stage three (§ 3.5) adopts a video-level temporal action detection formulation (§ 3.5.1) for the temporal trimming of action paths (§ 3.5.2). Next, we report the experimental validation of the proposed model in Section 3.6, from both a qualitative and a quantitative perspective. In Section 3.6.6, the implementation details are presented. Finally, we present the method's summary and limitations in Section 3.7.

3.2 Overview of action tube detection approach

Our approach is summarised in Figure 3.2. We train two pairs of Region Proposal Networks (RPN) [81] and Fast R-CNN [128] detection networks - one on RGB and the other on optical flow images [8]. For each pipeline, the RPN **(b)**, takes as input a video frame **(a)**, and generates a set of region proposals **(c)**, and their associated "actionness" [4] scores[5]. Next, a Fast R-CNN [81] detection network **(d)** takes as input the original video frame and a fraction of the region proposals generated by the RPN, and outputs a 'regressed' detection box and a softmax classification score for each input proposal, indicating the probability of an action class being present within the box. To merge appearance and motion cues, we fuse **(f)** the softmax scores from the appearance- and motion-based detection boxes **(e)** (§ 3.3.4). We find that this strategy significantly boosts detection accuracy.

After fusing the set of detections over the entire video, we identify sequences of frame regions most likely to be associated with a single action tube. Detection boxes in a tube need to display a high score for the considered action class, as well as a significant spatial overlap for consecutive detections. Class-specific action paths **(g)** spanning the whole video duration are generated via a Viterbi forward-backward pass (as in [8]). An additional second pass of dynamic programming is introduced to take care of temporal detection **(h)**. As a result, our action tubes are not constrained to span the entire video duration, as in [8]. Furthermore, extracting multiple paths allows our algorithm to account

[4]The term *actionness* [129] is used to denote the possibility of an action being present within a region proposal.

[5]A normalised score with softmax operator.

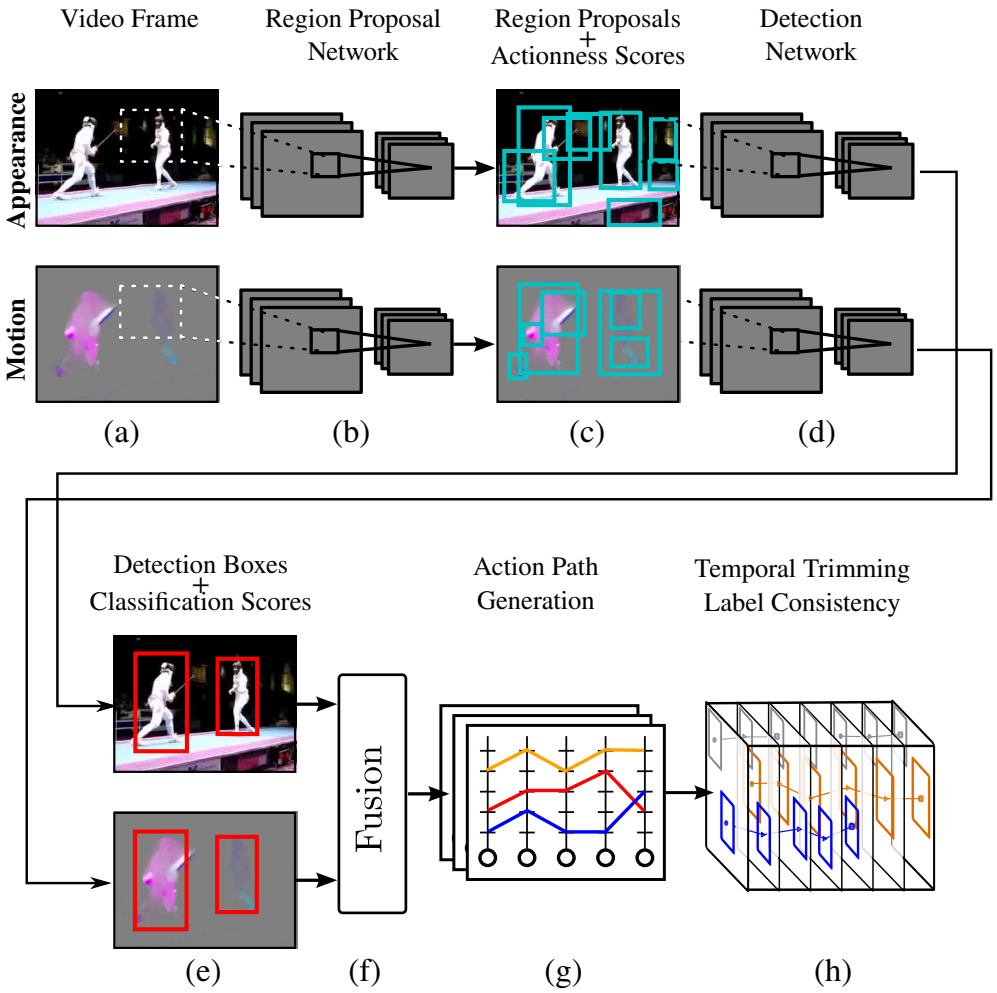


Figure 3.2: Overview of our two-stream deep action detection pipeline at test time. ***(a)*** *RGB and optical-flow images are passed to* ***(b)*** *two separate region proposal networks (RPNs) (§ 3.3.1).* ***(c)*** *Each network outputs its own region proposals with their associated "actionness" scores .* ***(d)*** *Each appearance/motion detection network (§ 3.3.2) takes as input the relevant image and RPN-generated region proposals, and* ***(e)*** *outputs detection boxes and softmax probability scores.* ***(f)*** *Appearance and motion based detections are fused (§ 3.3.4) and* ***(g)*** *linked up to generate class-specific action paths spanning the whole video (§ 3.4).* ***(h)*** *Finally the action paths are temporally trimmed to form action tubes (§ 3.5.2).*

for multiple co-occurring instances of the same action class.

3.2.1 Resulting contributions:

Although we makes use of existing RPN [81] and Fast R-CNN [128] architectures, this work proposes a radically new approach to spatiotemporal action detection which brings them together with a novel late fusion approach and an original action tube generation mechanism to dramatically improve accuracy and detection speed. Unlike [8, 13], in which appearance and motion infor-

mation are fused by combining fc7 (the fully connected layer 7) features, we propose a late fusion approach. Our novel fusion strategy boosts the confidence scores of the detection boxes based on their spatial overlaps and their class-specific softmax scores obtained from appearance and motion-based networks (Section 3.3.4).

A dynamic programming-based solution to the label consistency problem is introduced for the temporal trimming of each action tube. This contributes to a great extent to significantly improving detection performance (§ 3.6).

3.3 Frame-level detection framework

As outlined in Figure 3.2 our approach combines a region-proposal network (§ 3.3.1-Fig. 3.2(b)) with a detection network (§ 3.3.2-Fig. 3.2(d)), and fuses the resulting outputs (§ 3.3.4-Fig. 3.2(f)). These two components provide each a set of detection bounding boxes for each video frame, where each bounding box is associated with a classification score vector of length $C + 1$, including one for a 'background' class.

Each component of our system is described in detail below.

3.3.1 Region proposal network

To generate rectangular action region hypotheses in each video frame we adopt the Region Proposal Network (RPN) first published in [81], which is built on top of the last convolutional layer of the VGG-16 architecture by Simonyan and Zisserman [96]. To generate region proposals, this mini-network slides over the convolutional feature map outputted by the last layer, processing at each location a $n \times n$ spatial window and mapping it to a lower-dimensional feature vector (512-d for VGG-16). The feature vector is then passed to two fully connected layers: a box-regression layer and a box-classification layer.

During training, for each image location, k region proposals (also called 'anchors') [81] are generated. We consider those anchors having a high spatial intersection-over-union (IoU) with the ground truth boxes (IoU > 0.7) as positive examples, whilst those with IoU < 0.3 are considered as negative examples. Based on these training examples, the network's objective function is minimised using stochastic gradient descent (SGD), encouraging the prediction of both the probability of an anchor belonging to action or no-action category (a binary classification) and the 4 coordinates of the bounding box. Similiar to [81], smooth L1 loss function is used for regression and crossentrophy for classifaction objective.

3.3.2 Detection network

As the detection network we use a Fast R-CNN net [128] with a VGG-16 base architecture [96]. This takes the RPN-based region proposals (§ 3.3.1) and regresses a new set of bounding boxes for each action class with the associated classification scores. Each RPN-generated region proposal leads to C (number of classes) regressed bounding boxes with corresponding class scores.

Analogously to the RPN component, the detection network is also built upon the convolutional feature map outputted by the last layer of the VGG-16 network. It generates a feature vector for each proposal generated by RPN, which is again fed to two sibling fully-connected layers: a box-regression layer and a box-classification layer. Unlike what happens in RPNs, these layers produce C multi-class softmax scores and refined boxes (one for each action category) for each input region proposal.

3.3.3 CNN training strategy

We employ a variation on the training strategy of [81] to train both the RPN and the Fast R-CNN networks. Shaoqing *et al.* [81] suggested a 4-steps 'alternating training' algorithm in which in the first 2 steps, an RPN and Fast R-CNN nets are trained independently, while in the 3$^{\text{rd}}$ and 4$^{\text{th}}$ steps the two networks are fine-tuned with shared convolutional layers. In practice, we found empirically that the detection accuracy on UCF101-24 slightly decreases when using shared convolutional features, i.e., when fine-tuning the RPN and Fast-RCNN trained models obtained after the first two steps. As a result, we train the RPN and the Fast R-CNN networks independently following only the 1$^{\text{st}}$ and 2$^{\text{nd}}$ steps of [81], while neglecting the 3$^{\text{rd}}$ and 4$^{\text{th}}$ steps suggested by [81].

3.3.4 Fusion of appearance and motion cues

In a work by Redmon *et al.* [130], the authors combine the outputs from Fast R-CNN [128] and YOLO [130] (You Only Look Once) object detection networks to reduce background detections and improve the overall detection quality. Inspired by their work, we use our motion-based detection network to improve the scores of the appearance-based detection net (see Figure 3.2(f)).

Let $\{\mathbf{b}_i^s\}$ and $\{\mathbf{b}_j^f\}$ denote the sets of detection boxes generated by the appearance- and motion-based detection networks, respectively, on a given test frame and for a specific action class c. Let $\mathbf{b}_{max}^f$ be the motion-based detection box with maximum overlap with a given appearance-based detection box $\mathbf{b}_i^s$. If this maximum overlap, quantified using the IoU, is above a given threshold τ,

we augment the softmax score $s_c(\mathbf{b}_i^s)$ of the appearance-based box as follows:

$$s_c^*(\mathbf{b}_i^s) = s_c(\mathbf{b}_i^s) + s_c(\mathbf{b}_{max}^f) \times IoU(\mathbf{b}_i^s, \mathbf{b}_{max}^f). \tag{3.1}$$

The second term adds to the existing score of the appearance-based detection box a proportion, equal to the amount of overlap, of the motion-based detection score. We set $\tau = 0.3$ by cross-validation on the training set. We also tried augmenting the softmax scores of the motion-based detection boxes as per their maximum IoU overlaps with the appearance-based detections, but this did not produce better results.

3.4 Building action paths.

3.4.1 Problem formulation

The output of our fusion stage (§ 3.3.4) is, for each video frame, a collection of detection boxes for each action category, together with their associated augmented classification scores (Equation 3.1). Detection boxes can then be linked up in time to identify video regions most likely to be associated with single or multiple action instances, which we term *action paths*.

Action paths are connected sequences of detection boxes in time, without interruptions, which span the entire video duration. They are obtained as solutions to an energy maximisation problem. A number of action-specific paths $\mathbf{p}_c = [\mathbf{b}_1, \ldots, \mathbf{b}_T]$, spanning the entire video length, are constructed by linking detection boxes over time in virtue of their class-specific scores and their spatial overlap.
For example, consider a video composed of 200 frames, in which an action is present between frame 52 and 87. The generated action paths, however, will span the entire video duration – i.e., each path starts at frame 1 and ends at frame 200. For those frames where the action is not present, the action path generation algorithm (§ 3.4.2) picks up suitable detections according to the energy defined for a particular path (Equation 3.2). Subsequently, these noisy detections are expected to be removed from each path during the temporal trimming step (§ 3.5.2).

Video-long action paths are illustrated in Figure 3.3(a).

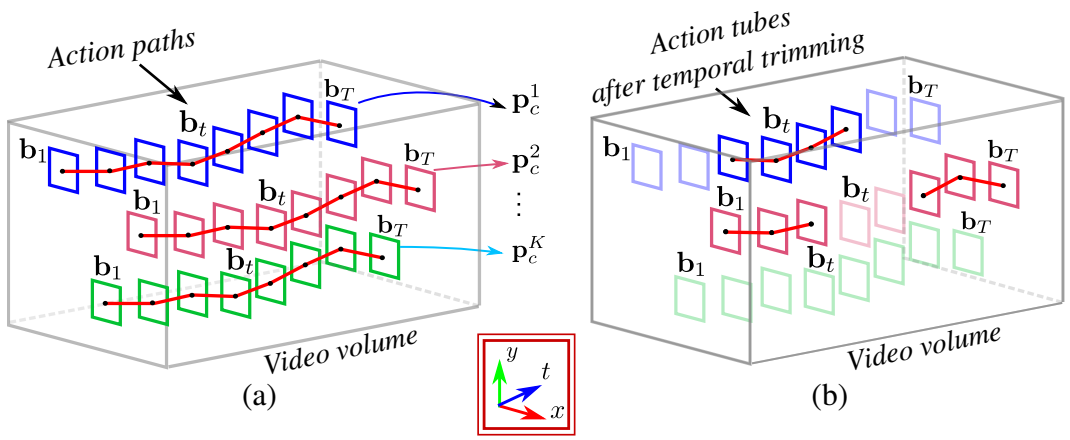


Figure 3.3: ***(a)*** *K class-specific action paths $\mathbf{p}_c$ are built by temporally linking a number of frame-level detections $\mathbf{b}_t$.* ***(b)*** *Action tubes are generated after applying temporal trimming on the K class-specific action paths. In this example, $\mathbf{p}_c^1$ is trimmed down to one action tube, $\mathbf{p}_c^2$ is trimmed to two action tubes, whereas $\mathbf{p}_c^K$ does not contain any actual action tube, hence it is entirely discarded.*

3.4.2 Optimisation formulation

We define the energy $E(\mathbf{p}_c)$ for a particular path $\mathbf{p}_c$ linking up detection boxes for class c across time to be a sum of unary and pairwise potentials:

$$E(\mathbf{p}_c) = \sum_{t=1}^{T} s_c^*(\mathbf{b}_t) + \lambda_o \sum_{t=2}^{T} \psi_o\left(\mathbf{b}_t, \mathbf{b}_{t-1}\right), \tag{3.2}$$

where $s_c^*(\mathbf{b}_t)$ denotes the augmented score (Equation 3.1) of detection $\mathbf{b}_t$, the overlap potential $\psi_o(\mathbf{b}_t, \mathbf{b}_{t-1})$ is the IoU of the two boxes $\mathbf{b}_t$ and $\mathbf{b}_{t-1}$, and λ_o is a scalar parameter weighting the relative importance of the pairwise term, λ_o is set to 1 after cross-validation. The value of the energy (Equation 3.2) is high for paths whose detection boxes score highly for the particular action category c, and for which consecutive detection boxes overlap significantly. We can find the path which maximises the energy,

$$\mathbf{p}_c^* = \text{argmax}_{\mathbf{p}_c}\ E(\mathbf{p}_c) \tag{3.3}$$

by simply applying the Viterbi algorithm used by [8].

Once an optimal path has been found, we remove all the detection boxes associated with it and recursively seek the next best action path. Extracting multiple paths allows our algorithm to account for multiple co-occurring instances of the same action class.

3.5 Temporal action detection

In this section we describe how a label consistency formulation can be applied to convert frame-level score vectors into a temporal action detection method (Section 3.5.1), as well as into a method designed to trim action paths to the actual temporal extent of the action instances there contained (Section 3.5.2), as illustrated in Figure 3.3(b).

3.5.1 Frame-level temporal action detection for the ActivityNet challenge

The optimisation formulation we used to perform temporal action detection in the context of the 2016 ActivityNet challenge is described here.

For that competition, we used the frame-level and video-level pre-computed CNN features provided by the organisers. Each video was first classified into one of the C classes using an ensemble of classifiers trained [20] on video-level features. We would only perform temporal trimming for the top-5 video-level labels. More details about video classification and features are given in Section 3.6.6 – here we are interested, in particular, in the label consistency formulation designed to solve the temporal action detection problem starting from the score vectors of the observed video frames. We trained a random forest [131] classifier on the frame-level features, which outputted a classification score of length C for each frame.

Formally, given the frame-level scores $\{s_t, t = 1, ..., T\}$ for a video of length T, we want to assign to each frame a binary label $l_t \in \{1, 0\}$ (where zero represents the 'background' or 'no-activity' class), which maximises:

$$E(L) = \sum_{t=1}^{T} s_t - \sum_{t=2}^{T} \psi_l \left(l_t, l_{t-1}\right), \tag{3.4}$$

where the pairwise potential ψ_l is defined as:

$$\psi_l(l_t, l_{t-1}) = \begin{cases} 0 & \text{if } l_t = l_{t-1} \\ \alpha & \text{otherwise,} \end{cases} \tag{3.5}$$

where α is a parameter which we set by cross-validation. This penalises labelling $L = \{l_1, ..., l_T\}$ which are not smooth, thus enforcing a piece-wise constant solution. All contiguous sub-sequences form the desired activity proposal (which can be as many as there are instances of activities). Each activity proposal is assigned a global score of S_a equal to the mean of the scores of its

constituting frames. This optimisation problem can be efficiently solved by dynamic programming [73].

3.5.2 Temporal trimming in action paths

We can address the temporal trimming of action paths with the same formulation as in Section 3.5.1.

As a result of the action path building formulation of Section 3.4, action-paths span the entire video duration, whereas human actions typically only occupy a fraction of it. Temporal trimming, therefore, becomes necessary. The dynamic programming solution of Equation 3.2 aims at extracting connected paths by penalising regions which do not overlap in time. As a result, however, not all detection boxes within a path exhibit strong action-class scores.

The goal here is to assign to every box $\mathbf{b}_t \in \mathbf{p}_c$ in an action path $\mathbf{p}_c$ a binary label $l_t \in \{c, 0\}$ (where zero represents the 'background' or 'no-action' class), subject to the conditions that the path's labelling $\mathbf{L}_{\mathbf{p}_c} = [l_1, l_2, \ldots, l_T]'$: i) is consistent with the unary scores (Equation 3.1); and ii) is smooth (no sudden jumps).
As in the previous pass, we may solve for the best labelling by maximising:

$$E(\mathbf{L})_{\mathbf{p}_c} = \underset{\mathbf{L}_{\mathbf{p_c}}}{\operatorname{argmax}} \left(\sum_{t=1}^{T} s_{l_t}(\mathbf{b}_t) - \lambda_l \sum_{t=2}^{T} \psi_l \left(l_t, l_{t-1} \right) \right), \tag{3.6}$$

where λ_l is a scalar parameter weighting the relative importance of the pairwise term. The pairwise potential ψ_l is defined to be:

$$\psi_l(l_t, l_{t-1}) = \begin{cases} 0 & \text{if } l_t = l_{t-1} \\ \alpha_c & \text{otherwise,} \end{cases} \tag{3.7}$$

where α_c is a class-specific constant parameter which we set by cross validation. As each action category has its own short-term and long-range motion dynamics, cross-validating α_c for each action class separately improves the temporal detection accuracy (§ 3.6.4). Equation 3.7 is the standard Potts model which penalises labelling that are not smooth, thus enforcing a piece-wise constant solution. Again, we solve (Equation 3.2) using the Viterbi algorithm.

All contiguous sub-sequences of the retained action paths $\mathbf{p}_c$ associated with category label c constitute our action tubes. As a result, one or more distinct action tubes spanning arbitrary temporal intervals may be found in each video for each action class c. Finally, each action tube is assigned a global score equal to the mean of the top k augmented class scores (Equation 3.3.4) of its

constituting detection boxes.

If an action occurs multiple times in a video, then each occurrence of that action represents a different action instance which is encoded by a different action tube. Figure 3.3(b) illustrates the action tube generation step in which K action tubes are obtained by applying temporal trimming on the K class-specific action paths.

3.6 Experimental validation

We evaluate our method on two separate challenging problems: i) the temporal action detection task on the ActivityNet dataset in the context of the 2016 challenge (§ 3.6.1), in order to assess the strength of the label consistency formulation introduced in Section 3.5; ii) the spatiotemporal action detection task (§ 3.6.2) on the UCF101-24, JHMDB-21-21 and LIRIS-HARL datasets, which evaluates our overall action tube detection framework (§ 3.2).

The detailed information about evaluation benchmarks is given in Section 2.9 in Chapter 2. In the following we show the results of our method on different thresholds to understand the performance under gradually more difficult tasks. The mAP metric, when used to evaluate the detection of tubes in videos, is called video-mAP. In our case, video-mAP and mAP are used interchangeably. More information about evaluation meteric can be found in Section 2.10.

T-IoU threshold $\delta =$	0.1	0.2	0.3	0.4	0.5
Validation-Set Caba *et al.* [28]	12.5%	11.9%	11.1%	10.4%	09.7%
Validation-Set Montes *et al.* [28]	–	–	–	–	22.5
Validation-Set proposed	52.1%	47.9%	43.5%	39.2%	34.5%
Testing-Set Singh *et al.* [68]	–	–	–	–	28.8%
Testing-Set proposed	–	–	–	–	36.4%

Table 3.1: Activity detection performance results [20] on validation and testing set. The quantity δ is the Temporal Intersection over Union (T-IoU) threshold.

3.6.1 Temporal detection performance comparison

Table 3.1 show the results obtained using the temporal label smoothing framework proposed in Section 3.5. Our method show superior performance than the baseline method of Caba *et al.* [28]. Note that we used the same pre-computed features as Caba *et al.* [28] to build our temporal action detection framework. It clearly shows that temporal detection formulation presented in Section 3.5 is effective, hence, its adoption for trimming the action paths in Section 3.5.2 is natural.

Dataset	UCF101-24			J-HMDB-21		LIRIS-HARL	
ST-IoU threshold δ	0.05	0.2	0.5	0.2	0.5	0.2	0.5
FAP [122]	42.80	–	–	–	–	–	–
ActionTube [8]	–	–	–	–	53.3	–	–
Wang *et al.* [132]	–	–	–	–	56.4	–	–
STMH [13]	54.28	46.77	–	63.1	60.7	–	–
Our (appearance stream)	67.56	56.55	30.64	52.94	51.34	41.94	20.43
Our (optical flow stream)	65.19	55.68	27.84	69.59	67.90	46.58	19.28
Our (fused)	**78.85**	**66.36**	**34.82**	**72.63**	**71.50**	**49.10**	**21.36**

Table 3.2: Quantitative action detection results (mAP) [14] on UCF101-24, J-HMDB-21 and LIRIS-HARL datasets.

3.6.2 Spatiotemporal detection performance comparison

Table 3.2 presents the results we obtained on UCF101-24, J-HMDB-21, LIRIS-HARL datasets. It contains a comparison with the previous state-of-the-art paper [8, 13, 122, 132] on each dataset. Next, we discuss the performance of our proposed method on each individual dataset. It is important to note that our fusion strategy (§ 3.3.4) helps improve the performance of our method consistently on all datasets – see the last three rows in Table 3.2.

Performance comparison on UCF101-24 We achieve an mAP of 66.36% compared to the 46.77% reported by [13] (a 20% gain), at the standard threshold of $\delta = 0.2$. At a threshold of $\delta = 0.4$ we still get a high score of 45.24%, (comparable to 46.77% [13] at $\delta = 0.2$). Note that we are the first to report results on UCF101-24 up to $\delta = 0.6$, which attests the robustness of our approach to more accurate detection requirements. Although our separate appearance- and motion-based detection pipelines already outperform the state-of-the-art (Table 3.2), their combination (§ 3.3.4) delivers a significant performance increase.

Some representative example results from UCF101-24 are shown in Figure 3.4. Our method can detect several (more than 2) action instances concurrently, as shown in Figure 3.1, in which three concurrent instances and in total of six action instances is detected correctly. Quantitatively, our method achieves class-specific video AP (average precision in %) of 88.0, 83.0 and 62.5 on the UCF101-24 action categories *'fencing'*, *'salsa spin'* and *'ice dancing'*, respectively, which all concern multiple inherently co-occurring action instances.

Performance comparison on J-HMDB-21 The results we obtained on J-HMDB-21 are presented in the second part of Table 3.2. Our method again outperforms the state-of-the-art, with an mAP increase of 18% and 11% at $\delta = 0.5$



Figure 3.4: Sample action detection results [14] on UCF101-24. Ground-truth boxes are in green, detection boxes in red. The top two rows show correct detections, the bottom one contains examples of more mixed results. In the second row right most frame, 3 out of 4 'Fencing' instances are nevertheless correctly detected.

as compared to [8] and [13], respectively. Note that our motion-based detection pipeline alone exhibits superior results, and when combined with appearance-based detections leads to a further improvement of 4% at $\delta = 0.5$.
These results confirm the high precision of the detections - a large portion of the detection boxes have high IoU overlap with the ground truth boxes, a feature due to the superior quality of RPN-based region proposals as opposed to Selective Search's (a direct comparison is provided in Section 3.6.3). Sample detections on J-HMDB-21 are shown in Figure 3.5.

Performance comparison on LIRIS-HARL LIRIS HARL allows us to demonstrate the efficacy of our approach on temporally long untrimmed videos with co-occurring actions. We also report in Table 3.2 the mAP scores obtained by the appearance, motion and the fusion detection models, respectively (note that there is no prior state of the art to report in this case). Again, we can observe an improvement of 7% mAP at $\delta = 0.2$ associated with our fusion strategy. Qualitative results on LIRIS reported in Figure 3.6 show how our method can handle co-occurring action instances of different action classes.

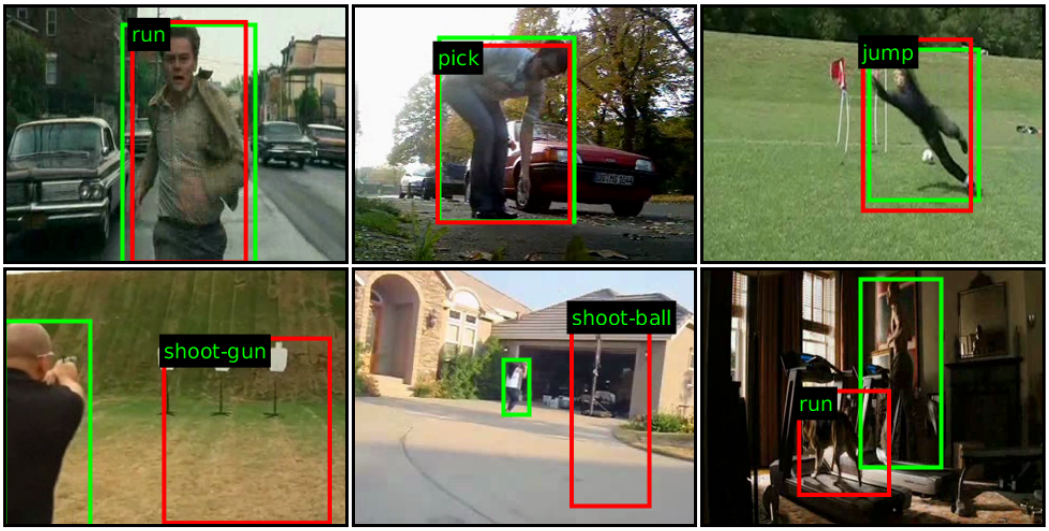


Figure 3.5: Sample action detection results [14] on JHMDB-21. Top row: accurate detection examples. Bottom row: mis-detection examples.

Similarly to what happens with the mAP metric, the F1 score can be computed using precision and recall at a particular threshold. The F1 score was used as an evaluation metric for the LIRIS-HARL dataset to set the benchmark. Our method achieves an F1 score of 0.58 as compared with 0.05 and 0.03 obtained by the two benchmark approaches of VPULABUAM-13 [133] and IACAS-51 [134] respectively.

3.6.3 Comparative analysis of region proposal quality

To understand the importance of region proposal generation, we also analyse the quality of Selective Search vs that of RPN (§ 3.3.1) region proposals using the Recall-to-IoU measure [81]. We first extract Selective Search (SS) boxes (approximately 1000 boxes/frame) and RPN-based detection boxes (300 boxes/frame) from our detection network on UCF101-24 test-split-1. We also apply a constraint on the RPN-based proposals by applying a threshold to their class-specific softmax probability scores s_c, and only considering those proposals with $s_c \geq 0.2$. For each UCF101-24 action category we compute the recall of these proposals at different threshold values. Even with a relatively smaller number of proposals and the additional constraint on the classification probability score, RPN-based proposals exhibit much better recall values than SS-based boxes as depicted in Figure 3.7.

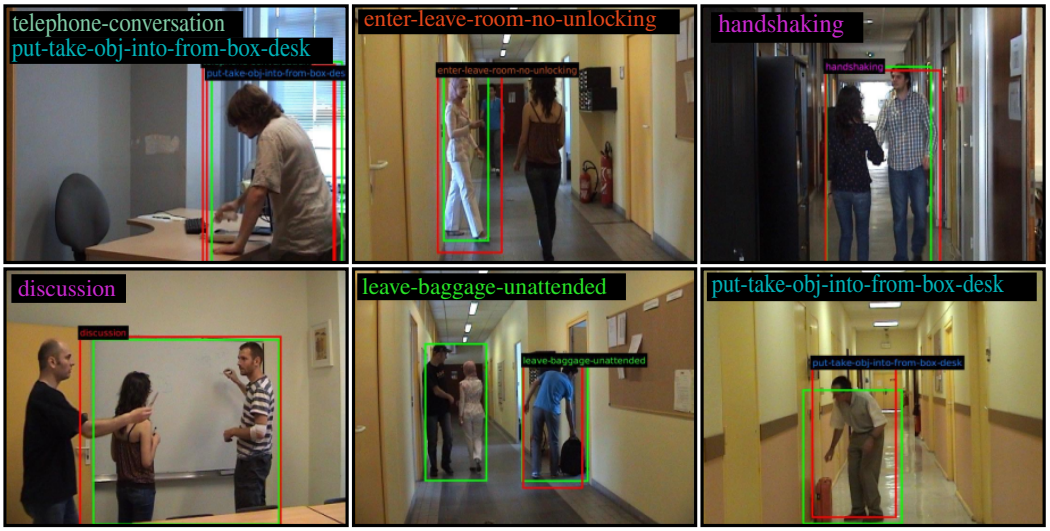


Figure 3.6: Sample action detection results [14] on LIRIS-HARL, some of which include single actions involving more than one person like 'handshaking' and 'discussion'. Top row: accurate detection examples. Bottom row: mis-detection examples.

3.6.4 Impact of label smoothing

Along with temporal action detection results shown on ActivityNet dataset in Section 3.6.1, we demonstrate the advantage of our temporal labele smoothing step (§ 3.5.2) on UCF101-24 dataset and LIRIS-HARL datasets. We shown results (mAP) using the action tubes generated by trimming action paths (§ 3.4.2) with our label smoothing method (§ 3.5.2) for action paths. Without the 2^{nd} (trimming) pass, performance decreases by 20%, highlighting the importance of temporal trimming in the construction of accurate action tubes.

First, we conduct an experiment where we evaluate the performance difference observed when we perform a temporal label smoothing (as in Section 3.5.2) of the action paths. We evaluate the action paths generated on LIRIS-HARL against the ground truth action tubes, i.e., without temporal trimming them using the method of Section 3.5.2: see the first row in Table 3.3. Similarly, we evaluate on LIRIS-HARL the action tubes generated using the label temporal smoothing formulation of Section 3.5.2: see the second row in Table 3.3. We can clearly see from Table 3.3 that proposed temporal label smoothing formulation helps significantly.

Proposed - no temporal smoothing (i.e., without § 3.5)	38.1	29.46	23.58	14.54	9.59
Proposed - with temporal smoothing (i.e., with § 3.5)	**54.18**	**49.10**	**35.91**	**28.03**	**21.36**

Table 3.3: Action detection results (mAP) on LIRIS-HARL with and without temporal triming (§ 3.5).

Next, we also conduct experiments to show the significance of the label smoothing step with the help of parameter class-specific α_c. More specifi-

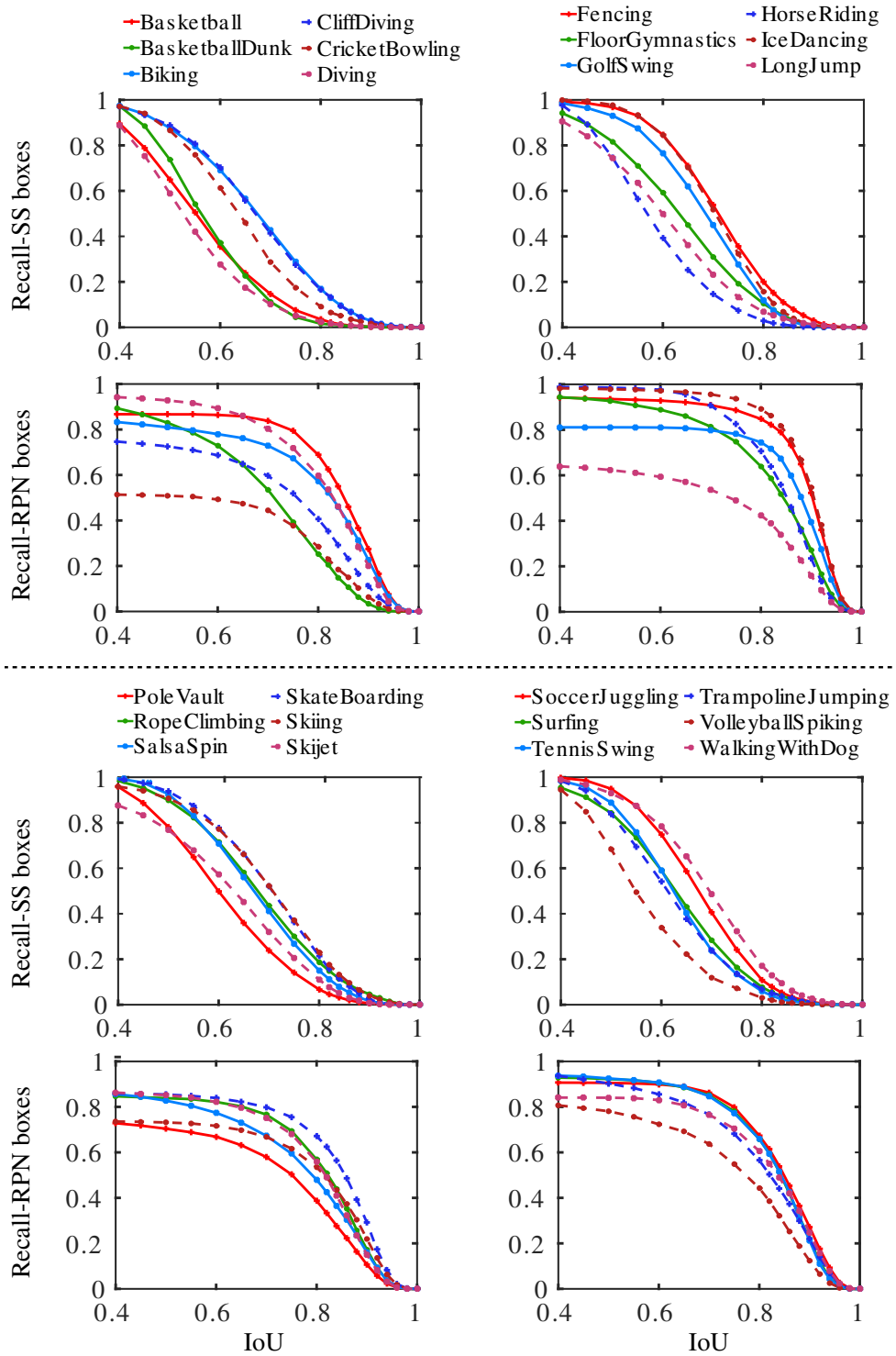


Figure 3.7: Performance comparison between Selective Search (SS) and RPN-based region proposals on 24 action classes of UCF101-24 dataset. 1^{st} & 3^{rd} rows: recall vs. IoU curve for Selective Search based region proposals; 2^{nd} & 4^{th} rows: recall vs. IoU curve for RPN-based region proposals.

cally, we show that class-specific α_c values help smoothing the action paths for each action category independently, resulting in an overall performance boost in the spatiotemporal detection accuracy. First, we generate detection results

on UCF101-24 test set (split-1) by setting the constant parameter $\alpha_c = 0$ for each action category. Then, we use a fixed value of $\alpha_c = 3$. Lastly, we use the cross-validated class-specific α_c values and again generated detection results. In our experiment, we set the spatiotemporal IoU threshold to $\delta = 0.2$.
Table 3.4 presents the results for the following three cases: **(a)** detection results obtained by setting $\alpha_c = 0$ for all action classes; **(b)** detection results generated using a fixed value of α_c for all classes; **(c)** detection results obtained using the cross validated class-specific α_c values. It shows that α_c should be greater than 0 to force piece-wise constant results, otherwise the resulting labelling path would just track the score values over time.

	mAP
$\alpha_c = 0$	60.77
$\alpha_c = 3$	66.03
class-specific α_c	**66.36**

Table 3.4: Spatiotemporal action detection results (mAP) on UCF101-24 using three different sets of α_c values.

3.6.5 Computing time analysis for testing

We think useful to compare the detection speed at test time of the combined region proposal generation and CNN feature extraction approach used in ([8, 13]) with our neural-net based, single stage action proposal and classification pipeline on the J-HMDB-21 dataset: see Table 3.5. We find our method to be 10× faster than [8] and 5× faster than [13], with a mean of 113.52 [8], 52.23 [13] and 10.89 (ours) seconds per video, averaged over all the videos in J-HMDB-21 split1.

We exclude our 2^{nd} label smoothing step (§ 3.5.2) due to the fact that J-HMDB-21 video clips do not require temporal trimming. The action path building (§ 3.4.2) step and optical flow based 'motion frame' generation steps are common to [8] and our pipeline – we thus exclude these steps as well in our comparison. We compare the computation times required for the region proposal generation and CNN feature extraction steps of [8, 13] with our RPN and detection nets computation times. Table 3.5 shows the time required for each step. The reported computation time is averaged over all the videos in the J-HMDB-21 test split1. The time is in second per video clip. It takes 10 seconds on an average to process a video by detection framework. On the other side of computation, we can say that it takes 200 milliseconds per frame to run detection framework. All the Experimental results were generated using

Table 3.5: Test time detection speed comparison on J-HMDB-21 with [8, 13].

ActionTube [8], STMH [13]	Average time (Sec./video)
Selective Search [8] / EdgeBoxes [13]	68.10 / 6.81
CNN feature extraction	45.42
Average detection time	113.52 [8] / 52.23 [13]

Ours	Average time(Sec./video)
RPN proposal generation	4.08
Detection network	6.81
Average detection time	**10.89**

a desktop computer with an Intel Xeon CPU@3.20GHz and NVIDIA Titan X GPU.

Our method is at least 10× faster than [8] and 5× than [13] for those steps we compared with (see Table 3.5).

It is important to note that our adoption of Faster-RCNN [128] for action tube detection does not only provide improvement in performance (see Table 3.2) but also a simpler and faster solution to the problem. We do away with the expensive sliding window-based approach in [13] for temporal detection in action paths, which is replaced by our efficent [73] label smoothing formulation.

When combined, our efficient label smoothing step and the use of a faster detection network produce an elegant, simple, and efficient framework for action tube detection.

3.6.6 Implementation details

Training data preparation We divided the UCF101-24 train split one into two subsets. The first subset consists of 70% (1605 videos $\sim 240k$ frames) of the training videos from UCF101-24 train split one, whereas the second subset contains the remaining 30% (688 videos) . We selected the videos uniformly at random for each action class, and trained the RPN and Fast R-CNN networks using the first subset, while the second subset was used as a validation set for CNN training. For the J-HMDB-21 and LIRIS HARL D2 datasets we used the original training sets provided by the authors [10, 135].

CNN weight initialisation The RPN and Fast R-CNN networks [81] were initialised with weights from a model pre-trained on ImageNet dataset [136].

CNN solver configuration setting. For UCF101-24, we trained both RPN and Fast R-CNN for $320k$ iterations. For the first $240k$ iterations we used a learning rate 0.001, while for the remaining $80k$ iterations a learning rate of 0.0001 was set. For both the J-HMDB-21 and the LIRIS-HARL datasets, we trained both RPN and Fast R-CNN networks for $180k$ iterations. For the first $120k$ iterations a learning rate of 0.001 was used - for the remaining $60k$ iterations, we set the learning rate to 0.0001. The momentum was set to a constant value of 0.9, while weight decay was fixed to 0.0005.

Stochastic Gradient Descent mini-batch size. We selected an SGD mini-batch size of 256 for RPN, and of 128 for Fast R-CNN training.

CNN training. First, we trained an RPN network with either a set of RGB or optical flow-based training video frames. At each training iteration, the RPN would take as input a video frame and its associated ground truth bounding boxes. Once the RPN net was trained, we used the trained model to extract frame-level region proposals. A trained RPN net outputs a set of region proposals (around $16k$ to $17k$) per frame and their associated "actionness" scores. We then filtered these region proposals using non-maximal suppression (NMS) and selected top $2k$ proposals based on their "actionness" scores. These top $2k$ region proposals along with the frame and its ground truth boxes were then passed to a Fast R-CNN for training.

CNN testing. Once training both RPN and Fast R-CNN networks, we extracted region proposals from test video frames using the learnt RPN model. Similarly to what done in the training stage, we filtered the region proposals using NMS - however, at test time, we chose the top 300 region proposals and passed them to the Fast R-CNN network to obtain the final detection boxes: a set of $300 \times C$ regressed boxes and their associated softmax probability scores (where C is the number of ground truth action categories in a given dataset). For each action category, we first filtered the detection boxes using NMS and then selected the top 5 boxes per frame based on their softmax probability scores. We used an NMS threshold of 0.7 to filter the RPN-generated region proposals, and a threshold of 0.3 when filtering the Fast R-CNN detection boxes.

Implementation details for temporal detection on Activitynet. In this case we used the features provided on ActivityNet's [28] web page[6]. More specifically, these video-level features comprised of two sets of features. First, Ima-

[6] `http://activity-net.org/challenges/2016/download.html`

geNetShuffle features, which are video-level features generated by [137] using a Google inception net (GoogLeNet [138]). Second, Motion Boundary Histogram (MBH) features, which are generated with the aid of the improved trajectories [40] executable[7]. Video level features were passed to an SVM classifier for video classification. Finally, C3D features were generated at 2 frames per second using a C3D network [53] with a temporal resolution of 16 frames. The frame-level features were then used for frame-level classification with a random forest classifier [131] to produce frame-level scores. The frame-level scores of top 5 video-level categories were fed to the temporal detection formulation described in Section 3.5 to obtain temporal action detection results. We refer to [20] for more details. The working code is available on our public GitHub repository[8].

3.7 Summary and limitations

Summary: The superior performance of the proposed method is due to a number of reasons. 1) Instead of using unsupervised region proposal algorithms as in [89, 139], our pipeline takes advantage of a supervised RPN-based region proposal approach which exhibits better recall values than [139] (see Section 3.6.3). 2) Our fusion technique (Section 3.3.4) improves the mAP score (over the individual appearance or motion models) by 9.4% on the UCF101-24, J-HMDB-21 and LIRIS HARL datasets respectively, as shown in Section 3.6.2 and Table 3.2. 3) Temporal detection via label consistency is responsible for significant improvements in mAP, quantifiable at around 20% on LIRIS HARL and at 6% on UCF-101: see Section 3.6.4.

Qualitative results are also provided in the supplementary video [9] and on the project's web page[10], where the code has also been made available.

Limitations Although the proposed method produces remarkable results compared to the previous state-of-the-art, while also being more efficient, it still suffers from three major limitations, which are described below.

Firstly, dynamic programming is used to solve the optimisation problem formulation of both action path building (§ 3.4) and temporal trimming (§ 3.5) of action paths into action tubes. A dynamic programming-based solution requires various iterative passes. Hence, it requires observations from the entire video to be at hand, and as a result is intrisically offline in nature, i.e., it cannot process videos in a streaming fashion.

[7] `http://lear.inrialpes.fr/people/wang/improved_trajectories`
[8] `https://github.com/gurkirt/actNet-inAct`
[9] `https://www.youtube.com/embed/vBZsTgjhWaQ`
[10] `http://sahasuman.bitbucket.io/bmvc2016`

Secondly, even though the proposed method is more efficient than other approaches, it cannot still perform in real time at test time. One of the bottleneck is the two-step (RPN and detection) frame-level detection framework (§ 3.3), which requires 200ms (§ 3.6.5) to process a single frame. Another bottleneck is the computation time required for dense optical flow estimation [22], which again does not allow real time processing.

Thirdly, in the approach presented in this chapter the temporal reasoning is limited to the tube construction method (§ 3.4) and the optical flow stream (§ 4.5). The detection network, on the other hand, does not involve any temporal reasoning either at training or at testing time. We think it is a limitation of similar frame-level methods not being able to learn to perform temporal reasoning implicitly.

Looking ahead The limitations explained above provides us with an opportunity to devise a new way to tackle action detection problems. In the next chapter, we propose to overcome the first two limitations with the help of an online, real-time action detection method. The online, real-time processing in an action detection system makes it applicable to a wide variety of problems and applications, such as human-robot interaction, self-driving vehicles, autonomous robotic surgery. We will cover these aspects in the next chapter. Later on, in Chapter 5, we will propose solutions to enhance the temporal reasoning for predicting longer tubes directly from the set of input frames.

Chapter 4

Online and Real-time Action Tube Detection

4.1 Introduction

It has been observed, both in the previous chapter (Chapter 3) and in other works [13, 14, 90], that detecting human action tubes in unconstrained videos is challenging, owing to their deformable, spatiotemporal nature. The problem is made even harder if the detection is to be performed in an online setting and at real-time speed. Despite the performance of state-of-the-art spatiotemporal action detection systems (as described in the previous chapter and in [14,90]) being far from real-time, current systems also assume that the entire video (taken as a 3D block of pixels) is available ahead of time to detect action instances. As previously explained in the limitation Section 3.7 of the previous chapter, in order for such a detector to be applicable in real-world scenarios, such as video surveillance and human-robot interaction, video frames need to be processed in real-time and in an online fashion. Moreover, the action detection system needs to generate action tubes in an incremental and online fashion, as each new frame is captured.

The results from Chapter 3 show that supervised region proposal generation is crucial to improve detection accuracy. Although, [14,90] exhibit the excellent spatiotemporal action detection accuracy (§ 3.6.2), their test time detection strategy involves using computationally expensive optical flow frames [22]. In addition, these approaches require a two step process, composed by a region proposal network (RPN) [81] and an R-CNN formulation [127], limiting real-time deployment. Also, [14, 90] both employ offline tube generation methods which process the entire video in two passes: the first pass links up detection boxes into paths which stretch from start to end of the video; the second pass is

there to temporally trim the video-long constructed paths down to proper action tubes. Finally, these methods' mAP at high detection overlap values remains disappointingly low, and incompatible with many real-world applications.

4.1.1 An online, real-time framework

In this chapter, we illustrate an online framework, originally proposed in [9] and outlined in Figure 4.1, which overcomes all the above limitations. More specifically we propose three fundamental changes compared to the (then) existing action tube detection pipelines [14, 90] to achieve real-time processing and overcome the offline nature of the tube building process.

Firstly, the pipeline takes advantage of the SSD (Single Shot MultiBox Detector) object detector [21] to address issues with accuracy and speed in previous [81] frame-level detection networks. This is possible as SSD eliminates the region proposal generation step and is a single-stage, end-to-end trainable detection network.

Secondly, to leverage the performance of SSD in an online fashion, we design a novel single-pass online tube building method which leads to both superior accuracies (compared to [13, 14, 90]), especially at realistic detection precision, and better real-time detection speed. Unlike previous tube-generation approaches [8, 13, 14, 90] our algorithm works in an online fashion as tubes are updated frame by frame, together with their overall action-specific scores and labels. As soon as non-real-time optical flow [22] is replaced by a less accurate (but real-time) optical flow algorithm [23], the resulting system performs in real-time (28fps), with just a little performance degradation, an essential feature for real-world applications.

Lastly, we show that we can replace non-real-time optical flow [22] with less accurate but real-time optical flow proposed by [23] in two stream setting (shown in 4.1) without sacrificing much performance. Although a CNN trained on real-time optical flow [23] does not lead to the same performance as to the network trained on more accurate optical flow [22] by itself, but when used in a two-stream setting (in which detections from the RBG and the OF streams are fused) it can provide a similar boost as provided by an accurate optical flow [22] network in parallel to the appearance stream.

4.1.2 Extension to early label predictions

The incremental nature of our system makes it possible to accurately foresee the class label of an entire test video and detect action instances within it by

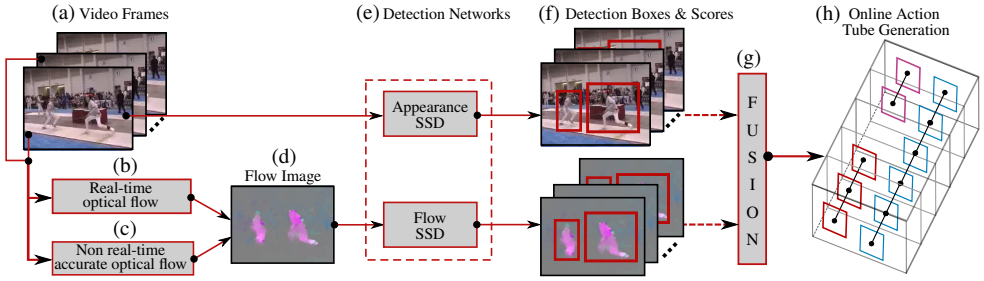


Figure 4.1: At test time, the input to the framework is a sequence of RGB video frames ***(a)****. A real-time optical flow (OF) algorithm* ***(b)*** *[23] takes the consecutive RGB frames as input to produce flow images* ***(d)****. As an option,* ***(c)*** *a more accurate optical flow algorithm [22] can be used (although not in real time).* ***(e)*** *RGB and OF images are fed to two separate SSD detection [21] networks (§ 4.3).* ***(f)*** *Each network outputs a set of detection boxes along with their class-specific confidence scores.* ***(g)*** *Appearance and flow detections are fused (§ 4.3.2). Finally* ***(h)****, multiple action tubes are built up in an online fashion by associating current detections with partial tubes (§ 4.4).*

just observing a small fraction of frames (the *early action prediction and detection* problem, which we described in the Introduction). Such a system was proposed by Soomro *et al.* [108], who showed that both action prediction and online detection performance improve over time as more and more video frames become available. Using [108] as a baseline, we report here new state-of-the-art results on the temporally trimmed J-HMDB-21 videos. Furthermore, compared to Soomro *et al.* [108], we are able to demonstrate action prediction and detection capabilities from partially observed *untrimmed* streaming videos on the challenging UCF101-24 dataset, while retaining real-time detection speeds.

In summary, we demonstrate that our online approach:

- significantly outperforms current offline methods, especially on realistic detection thresholds of 0.5 or greater;
- is capable of superior early action prediction performance compared to the existing baseline [108];
- achieves a real-time detection speed (up to 40fps), that is 5 to 6 times faster than previous works.

Related publications: The work presented in this chapter appeared in ICCV 2017 [9]. The dissertation author was the primary investigator in [9]. Our code is also available online[1]. Our statistics show that the code has been forked 60 times and stared 188 times in GitHub.

Outline: This chapter is organised as follows. We start by presenting an overview of our action tube detection approach in Section 4.2. Next, we de-

[1] `https://github.com/gurkirt/realtime-action-detection`

scribe in detail the three steps we introduced in this new online action tube detection framework. We first describe the integration of the Single-Shot Detector network [21] in Section 4.3, which is used to predict detection boxes and class-specific confidence scores for appearance and flow frames independently. Secondly, we describe our proposed novel online tube construction algorithm in Section 4.4. The third step (§ 4.5) is where we describe the efficient computation of optical flow. Section 4.6 illustrates, in particular, how the online tube construction algorithm can be used for early action label prediction. Next, we report the experimental validation of the proposed model (Section 4.7) from both a qualitative and a quantitative perspective. Finally, we present a summary and the approach's limitations in Section 4.8.

4.2 Overview of the approach

The proposed framework is outlined in Figure 4.1. We train two SSD detection networks [21]: one on RGB frames (appearance), the other on optical-flow images [8]. At test time, the input to the framework is a sequence of RGB video frames (Fig. 4.1(a)). A real-time optical flow algorithm [23] takes the RGB frames as input to produce flow images (Fig. 4.1(d)). As an option, more accurate OF (Fig. 4.1(c)) images can be computed in a non real-time setting [22] (but still incrementally). For each pipeline, the related detection network (Fig. 4.1(b)) takes as input an RGB (or OF) frame, and outputs a set of 'regressed' detection boxes (Fig. 4.1(f)), together with a set of class-specific confidence scores for each box indicating the likelihood of an action class being present there. We then we merge (Fig. 4.1(g)) the class-specific confidence scores associated with the flow-based and the appearance-based detection boxes, as this significantly boosts detection accuracy. Finally, sequences of detection boxes are linked up to form action-specific tubes in a completely online, incremental fashion (Fig. 4.1(h)).

The resulting framework is the first method to perform action tube detection in an online and real-time fashion. Unlike those in [8], the resulting action tubes are not constrained to span the entire video duration. Unlike what happens in [13, 14, 90], tubes are built in an online fashion allowing for early action label prediction and detection. Usage of real-time optical flow [23], and SSD detection networks [21] allows us ot process videos in streaming (online and real-time) fashion.

4.2.1 Resulting contributions

We present a holistic framework for the real-time, online spatial and temporal detection of multiple action instances in videos which:

- incorporates the SSD [21] neural architecture to improve the speed to predict frame-level detection boxes and the associated action class-specific confidence scores, in a single-stage regression and classification approach (§ 4.3);
- devises an original, greedy online algorithm capable of generating multiple action tubes incrementally (§ 4.4 - Fig. 4.1(g));
- provides early action class label predictions (§ 4.6) and online spatiotemporal detection (§ 4.4) from partially observed action instances in untrimmed videos;
- functions in real-time, while outperforming the previous (offline) state of the art on the untrimmed videos of UCF101-24 dataset.

Our framework is the first with a demonstrated ability to perform online spatiotemporal action detection.

4.3 Integrated detection network

We use a single-stage convolutional neural network (Fig. 4.1(e)) for bounding box prediction and classification, which follows an end-to-end trainable architecture proposed in [21]. The architecture unifies a number of functionalities in single CNN which are, in other action and object detectors, performed by separate components [8, 13, 14, 81], namely: (i) region proposal generation, (ii) bounding box prediction and (iii) estimation of class-specific confidence scores for the predicted boxes. This allows for relatively faster training and higher test time detection speeds.

4.3.1 Detection network design and training.

For our integrated detection network we adopt the network design and architecture of the SSD [21] object detector, with an input image size of 300×300. We do not use the 512×512 SSD architecture [21], as detection speed is much slower [21]. As in [21], we also use an ImageNet pretrained VGG16 network provided by [21] (`https://gist.github.com/weiliu89/2ed6e13bfd5b57cf81d6`). We adopt the training procedure described by [21] along with their publicly available code for network training (`https://github.com/weiliu89/caffe/tree/ssd`). We use a learning rate of 0.0005 for the appearance stream and of 0.0001 for the flow stream on UCF101-24, whereas that for JHMDB is set to 0.0001 for both appearance and flow.

4.3.2 Fusion of appearance and flow cues

The detection boxes generated by the appearance and flow detection networks (Fig. 4.1(f)) need to be merged to improve robustness and accuracy (Fig. 4.1(g)). We conducted experiments using two distinct fusion strategies.

Boost-fusion. As in the previous chapter, here we follow the approach in [14], with a minor modification. Firstly, we perform L-1 normalisation on the detection boxes' scores after fusion. Secondly, we retain any flow detection boxes for which an associated appearance based box was not found, as we found that discarding the boxes lowers the overall recall.

Fusion by taking the union-set. A different, effective fusion strategy consists in retaining the union $\{b_i^a\} \cup \{b_j^f\}$ of the two sets of appearance $\{b_i^a\}$ and flow $\{b_j^f\}$ detection boxes, respectively. The rationale is that in UCF101-24, for instance, several action classes (such as 'Biking', 'IceDancing', or 'SalsaSpin') have concurrent action instances in the majority of video clips: an increased number of detection boxes may so help to detect concurrent action instances.

4.4 Online action tube generation

4.4.1 Problem formulation

Given a set of detections at time $t = 1, \ldots, T$, for each given action class c, we seek the sets of consecutive detections (or *action tubes*) $\mathcal{T}_c = \{b_{t_s}, \ldots, b_{t_e}\}$ which, among all possible such collections, are more likely to constitute an action instance. This is done separately for each class, so that the results for class c do not influence those for other classes. We allow the number of tubes $n_c(t)$ to vary in time, within the constraint given by the number of available input detections. We allow action tubes to start or end at any given time. Finally, we require: (i) consecutive detections part of an action tube to have spatial overlap above a threshold λ; (ii) each class-specific detection to belong to a single action tube; (iii) the online update of the tubes' temporal labels.

Previous approaches to the problem [8, 14] would constrain tubes to span the entire video duration. In both [14] and [90], in addition, action paths are temporally trimmed to proper action tubes using a second pass of dynamic programming.

In opposition, we propose a simple but efficient online action tube generation algorithm which incrementally (frame by frame) builds multiple action tubes for each action class in parallel. Action tubes are treated as 'tracklets', as in multi-target tracking approaches [140]. We propose a greedy algorithm (4.4.2) similar

to [141, 142] for associating detection boxes in the upcoming frame with the current set of (partial) action tubes. Concurrently, each tube is temporally trimmed in an online temporal labelling (4.4.3) setting.

4.4.2 A novel greedy algorithm

The input to the algorithm is the fused frame-level detection boxes with their class-specific scores (Sec. 4.3.2). At each time step t, the top n class-specific detection boxes $\{b_c\}$ are selected by applying non-maximum suppression on a per-class basis. At the first frame of the video, $n_c(1) = n$ action tubes per class c are initialised using the n detection boxes at $t = 1$. The algorithm incrementally grows the tubes over time by adding one box at a time. The number of tubes $n_c(t)$ varies with time, as new tubes are added and/or old tubes are terminated.

At each time step, we sort the existing partial tubes so that the best tube can potentially match the best box from the set of detection boxes in the next frame t. Also, for each partial tube $\mathcal{T}_c^i$ at time $t-1$, we restrict the potential matches to detection boxes at time t whose IoU (Intersection over Union) with the last box of $\mathcal{T}_c^i$ is above a threshold λ. In this way, tubes cannot simply drift off, and they can be terminated if no matches are found for k consecutive frames. Finally, each newly updated tube is temporally trimmed by performing a binary labelling

Summarising, action tubes are constructed by applying the following 7 steps to every new frame at time t:

1. Execute steps 2 to 7 for each class c.
2. Sort the action tubes generated up to time $t-1$ in decreasing order, based on the mean of the class scores of the tube's member detection boxes.
3. **LOOP START:** $i = 1$ to $n_c(t-1)$ - traverse the sorted tube list.
4. Pick tube $\mathcal{T}_c^i$ from the list and find a matching box for it among the n class-specific detection boxes $\{b_c^j, j = 1, ..., n\}$ at frame t based on the following conditions:
   (a) for all $j = 1, ..., n$, if the IoU between the last box of tube $\mathcal{T}_c^i$ and the detection box b_c^j is greater than λ, then add it to a potential match list $\mathcal{B}^i$;
   (b) if the list of potential matches is not empty, $\mathcal{B}^i \neq \emptyset$, select the box b_c^{max} from $\mathcal{B}^i$ with the highest score for class c as the match, and remove it from the set of available detection boxes at time t;
   (c) if $\mathcal{B}^i = \emptyset$, retain the tube anyway, without adding any new detection box, unless more than k frames have passed with no match found for it.

5. Update the temporal labelling for tube $\mathcal{T}_c^i$ using the score $s(b_c^{max})$ of the selected box b_c^{max} (see § 4.4.3).
6. **LOOP END**
7. If any detection box is left unassigned, start a new tube at time t using this box.

In all our experiments, we set $\lambda = 0.1$, $n = 10$, and $k = 5$.

4.4.3 Temporal labelling

Although n action tubes per class are initialised at frame $t = 1$, we want all action specific tubes to be allowed to start and end at any arbitrary time points t_s and t_e. The online temporal relabelling step 5 in the above algorithm is designed to take care of this.

Similarly to [14, 73] and to Chapter 3's approach, each detection box b_r, $r = 1, ..., T$ in a tube $\mathcal{T}_c$, where T is the current duration of the tube and r is its temporal position within it, is assigned a binary label $l_r \in \{c, 0\}$, where c is the tube's class label and 0 denotes the background class. The temporal trimming of an action tube thus reduces to finding an optimal binary labelling $\mathbf{l} = \{l_1, ..., l_T\}$ for all the constituting bounding boxes. This can be achieved by maximising for each tube $\mathcal{T}_c$ the energy:

$$E(\mathbf{l}) = \sum_{r=1}^{T} s_{l_r}(b_r) - \alpha_l \sum_{r=2}^{T} \psi_l \left(l_r, l_{r-1}\right), \tag{4.1}$$

where $s_{l_r}(b_r) = s_c(b_r)$ if $l_r = c$, $1 - s_c(b_r)$ if $l_r = 0$, α_l is a scalar parameter, and the pairwise potential ψ_l is defined as: $\psi_l(l_r, l_{r-1}) = 0$ if $l_r = l_{r-1}$, $\psi_l(l_r, l_{r-1}) = \alpha_c$ otherwise.

Online Viterbi

The maximisation problem (4.1) can be solved by Viterbi dynamic programming [14,73]. An optimal labelling $\hat{\mathbf{l}}$ for a tube $\mathcal{T}_c$ can be generated by a Viterbi backward pass at any arbitrary time instant t in linear time. That, however, would require a backward pass all the way back to the start of each action path, which is expensive to perform for every action path at each time step. Instead, we keep track of the minimum cost edges from the past box-to-tube associations from the start of the tube up to $t - m$, which eliminates the requirement for an entire backward pass at each time step but only up to the previous m steps. In the following we used m equal to 5. This makes temporal labelling very efficient, and suitable to be used in an online fashion. This can be further

optimised for much longer videos by finding the coalescence point [143]. As stated in step 5 above, the temporal labelling of each tube is updated at each time step whenever a new box is added.

4.5 Optical flow computation

The input to our framework is a sequence of RGB images. As in prior work in action detection [8, 13, 14], we use a two-stream CNN approach [46] in which optical flow and appearance are processed in two parallel, distinct streams. As our aim is to perform action detection in real-time, we employ real-time optical flow (Fig. 4.1(b)) [23] to generate the flow images (Fig. 4.1(d)). As an option, one can compute optical flow more accurately (Fig. 4.1(c)), using Brox *et al.*'s [22] method. We thus train two different networks for the two OF algorithms, while at test time only one network is used depending on whether the focus is on speed rather than accuracy. Following the transfer learning approach on motion vectors of [100], we first train the SSD network on accurate flow results, to later transfer the learned weights to initialise those of the real-time OF network. Performance would degrade whenever transfer learning was not used.

4.6 Early action prediction

As for each test video multiple tubes are built incrementally at each time step t (§4.4), we can predict at any time instant the label of the whole video as the label of the current highest-scoring tube, where the score of a tube is defined as the mean of the tube boxes' individual detection scores:

$$\hat{c}(t) = \arg\max_{c} \left(\max_{\mathcal{T}_c} \frac{1}{T} \sum_{r=1}^{T} s(b_r) \right). \tag{4.2}$$

4.7 Experiments

We test our online framework on three separate challenging setups:
i) early action label prediction (§ 4.7.1);
ii) online spatiotemporal action detection over time (§ 4.7.2);
iii) an offline setting in which entire video is observed, so that the global spatiotemporal action detection performance can be assessed (§ 4.7.3), including a comparison with offline action detection methods.

In all settings we generate results by running our framework in five different 'modes':

1. *Appearance (A)* – only RGB video frames are processed by a single SSD;

2. *Real-time flow (RTF)* – optical flow images are computed in real-time [23] and fed to a single SSD;

3. *A+RTF*: both RGB and real-time optical flow images are processed by SSD in two separate streams;

4. *Accurate flow (AF)* optical flow images are computed as in [22], and

5. *A+AF*: both RGB and non-real-time optical flow frames [22] are used.

Modes 1), 2) and 3) run in real-time whereas modes 4) and 5)'s performances are non-real-time (while still working incrementally), due to the relatively higher computation time needed to generate accurate optical flow. Note that all experiments are conducted in a fully online fashion, using the proposed tube generation algorithm (§ 4.4).

In Section 4.7.4, we see how our online tube construction method contributes to the overall performance of our method. Next, we ablate the contribution of the different optical methods in Section 4.7.5. Lastly, we provide evidence of real-time capability in Section 6.6.4.

As in the previous chapter and in previous related works [13, 14, 90, 122] we evaluate our model on the UCF101-24 [1] and J-HMDB-21 [10] benchmarks. Please refer to Section 2.9 in Chapter 2 for more details on these dataset. We also used ActivityNet [6] dataset to evaluate temporal action detection performance, more details about the dataset are available in Section 2.9 in Chapter 2.

Evaluation metrics. For the early action label prediction (§ 4.7.1) and the online action detection (§ 4.7.2) tasks we follow the experimental setup of [108], and use the traditional detection metrics AUC (area under the curve) and mAP (mean average precision). We report performance as a function of *Video Observation Percentage*, i.e., with respect to the portion (%) of the entire video observed before predicting action label and location.

We also report a performance comparison with the relevant offline methods [13, 14, 90, 122] using the mean-average-precision (mAP) at various detection thresholds (δ), as explained in Section **??** in Chapter 3. Also, similarly to the evaluation protocol for object detection in the Microsoft COCO [144] dataset, we propose to use the *"average detection performance"* (avg-mAP) to

compare our performance with that of the previous state-of-the-art. To obtain the latter, we first compute the video-mAPs at higher IoU thresholds (δ) ranging $[0.5 : 0.05 : 0.95]$, and then take the average of these video-mAPs.

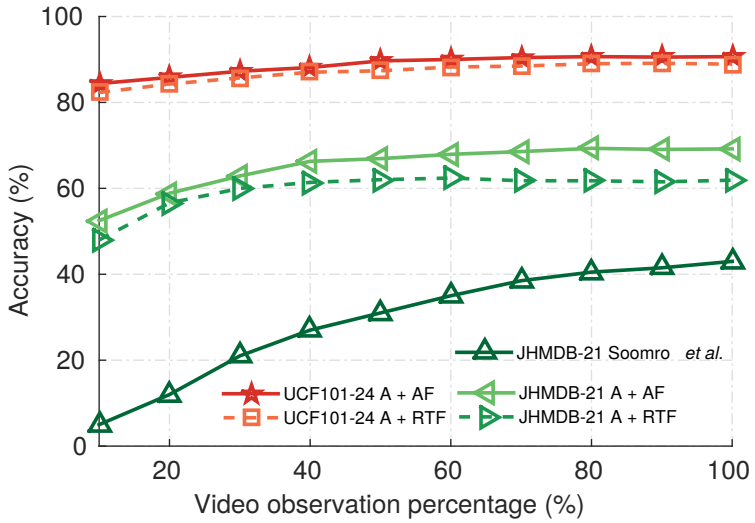


Figure 4.2: Early action label prediction results (accuracy %) over the UCF101-24 and J-HMDB-21 datasets.

4.7.1 Early action label prediction

Here, we need to drive the inference (obtain the video action label) before the action is over (see Section 4.6 for video-level score computation). Although action tubes are computed by our framework frame by frame, we sample them at 10 different time 'check-points' on each video, starting at 10% of the total number of video frames and with a step size of 10%. We use the union-set and boost fusion strategies (§ 4.3.2) for UCF101-24 and J-HMDB-21, respectively. Fig. 4.2 compares the early action prediction accuracy of our approach with that of [108], as a function of the portion (%) of video observed. Our method clearly demonstrates superior performance, as it is able to predict the actual video label by observing a very small portion of the entire video at a very initial stage. For instance, by observing only the initial 10% of the videos in J-HMDB-21, we are able to achieve a prediction accuracy of 48% as compared to 5% by Soomro *et al.* [108], which is, in fact, higher than the 43% accuracy achieved by [108] after observing the *entire* video. We do not run comparisons with the early action prediction work by Ma *et al.* [50] for they only show results on ActivityNet [28], as dataset which has only temporal annotations. The early prediction capability of our approach is a sub-product of its being online, as in [108]: thus, we only compare ourselves with Soomro *et al.* [108] re early action prediction results.

Compared to [108] we take one step further, and perform early label prediction on the untrimmed videos of UCF101-24 as well (see Fig. 4.2). It can be noted that our method performs much better on UCF101-24 than on J-HMBD-21 at the prediction task. This relatively higher performance may be attributed to the larger number of training examples, subject to more modes of variations, present in UCF101-24, which improves the generalisation ability of the model and prevents it from overfitting. Interestingly, we can observe that the performances of the real-time (A + RTF) and non-real-time (A + AF) modalities are quite similar, which suggests that accurate optical flow might be not so crucial for action classification on UCF101-24 dataset, which strongly supports our framework's ambition to accurately solve the problem in real-world situations in which real-time speed is required.

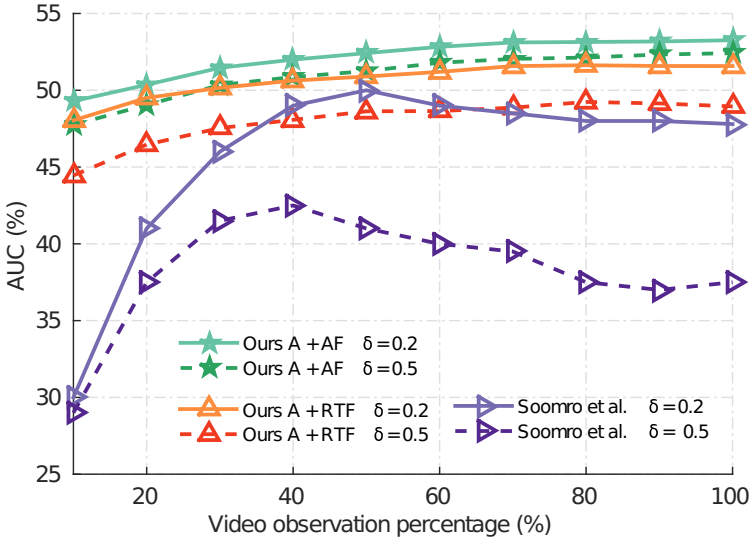


Figure 4.3: Online action detection results using the AUC (%) metric on J-HMDB-21, at IoU thresholds of $\delta = 0.2, 0.5$.

4.7.2 Spatiotemporal performance over time

Our action tubes are built incrementally and carry associated labels and scores at each time step. At any arbitrary time t, we can thus compute the spatiotemporal IoU between the tubes generated by our online algorithm and the ground truth tubes, up to time t.

Figure 4.3 plots the AUC curves against the observed portion of the video at different IoU thresholds ($\delta = 0.2$ and 0.5) for the proposed approach versus our competitor [108]. Our method outperforms [108] on online action detection by a large margin at all the IoU thresholds and video observation percentage. Notice that our online detection performance (Fig. 4.3) is a stable function of the video observation percentage, whereas Soomro *et al.* [108]'s method needs

some 'warm-up' time to reach stability, and its accuracy slightly decreases at the end. In addition, [108] only reports online spatial detection results on the temporally trimmed J-HMDB-21 test videos, and their approach lacks temporal detection capabilities.

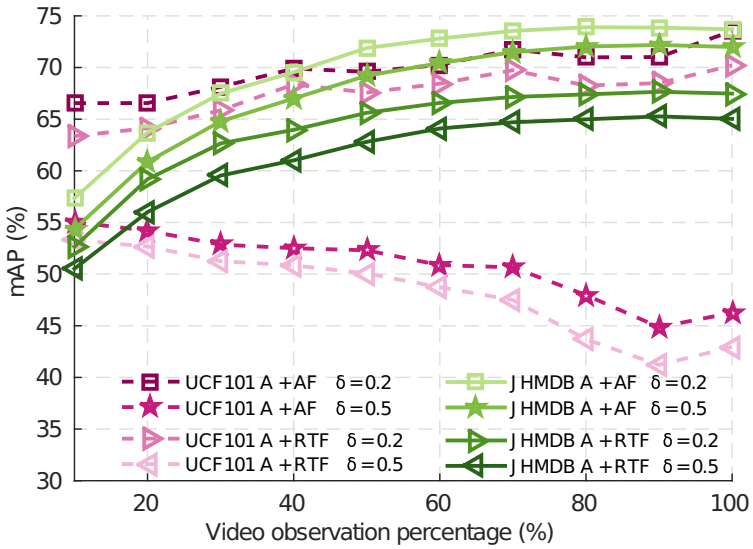


Figure 4.4: Action detection results using the mAP (%) metric on UCF101-24 and JHMDB-21, at IoU thresholds of $\delta = 0.2, 0.5$.

Our framework, instead, can perform online spatiotemporal detection: to demonstrate this, we present results on the temporally untrimmed UCF101-24 test videos as well. In Figure 4.4 we report online spatial-temporal detection results on UCF101-24 and JHMBD-21 using the standard mAP metric (not reported in [108]). Interestingly, for UCF101-24, at a relatively smaller IoU threshold ($\delta = 0.2$) the performance gradually increases over time as more video frames are observed, whereas at a higher IoU threshold ($\delta = 0.5$) it slightly degrades over time. A reason for this could be that UCF101-24 videos are temporally untrimmed and contain multiple action instances, so that accurate detection may be challenging at higher detection thresholds (e.g. $\delta = 0.5$). If temporal labelling is not very accurate, as required at high thresholds ($\delta = 0.5$), this might result in more false positives as the video progress, hence the observed drop in performance over time.

4.7.3 Global spatiotemporal performance

To demonstrate the strength of our online framework, we compare as well its absolute detection performances (avg-mAP after the video is fully observed) to those of the previous offline competitors [13, 14, 90, 122]. To ensure a fair comparison with [14] (tube construction method presented in the previous chapter),

Table 4.1: Action detection results (mAP) on UCF101-24 dataset in split1.

IoU threshold δ	0.2	0.5	0.75	avg-mAP
Yu *et al.* [122]‡	26.5	–	–	–
Weinzaepfel *et al.* [13]‡	46.8	–	–	–
Peng and Schmid [90]†	**73.5**	32.1	02.7	07.3
Saha *et al.* [14]†	66.6	36.4	07.9	14.4
Ours: Appearance (A)*	69.8	40.9	**15.5**	18.7
Ours: Real-time flow (RTF)*	42.5	13.9	00.5	03.3
Ours: A + RTF (boost-fusion)*	69.7	41.9	14.1	18.4
Ours: A + RTF (union-set)*	70.2	43.0	14.5	19.2
Ours: Accurate - flow (AF)**	63.7	30.8	02.8	11.0
Ours: A + AF (boost-fusion)**	73.0	44.0	14.1	19.2
Ours: A + AF (union-set)**	**73.5**	**46.3**	15.0	**20.4**
SSD+ [14] A + AF (union-set)†	71.7	43.3	13.2	18.6

‡ These methods were using different annotations to [14, 90] and ours. New annots are available at https://github.com/gurkirt/corrected-UCF101-Annots
* Incremental & real-time ** Incremental, non real-time † Offline

we evaluate their offline tube generation method using the detection of bounding boxes produced by the SSD net.

Improvement over the top performers. Results on UCF101-24 are reported in Table 4.1. In an online real-time setting we achieve an mAP of 70.2% compared to 66.6% reported by [14] at the standard IoU threshold of $\delta = 0.2$. In non-real time mode, we observe a further performance improvement of around 3.3%, leading to a 73.5% mAP, comparable to the 73.5 reported by the current top performer [90]. The similar performance of our method (A+AF) to [90] at $\delta = 0.2$ suggests that SSD and the multi-region adaptation of Faster-RCNN by [90] produce similar quality frame level detection boxes.

Performance under more realistic requirements. Our method significantly outperforms [14, 90] at more meaningful detection thresholds $\delta = 0.5$ or higher. For instance, we achieve a 46.2% mAP at $\delta = 0.5$ as opposed to 32.1% by [90] and 36.4% by [14], an improvement of **14%** and **9.8%**, respectively. This attests the superiority of our tube building algorithm when compared to those of [14, 90]. In fact, even in real time mode our pipeline (A + RTF) still performs better than both [14, 90] at $\delta = 0.5$ or higher.

It is important to note that, our proposed fusion method (*union-set-fusion*) significantly outperforms *boost-fusion* proposed by [14] on UCF101-24 dataset (see Table 4.1). UCF101-24 includes many co-occurring action instances, we can infer that the union-set fusion strategy improves the performance by providing a larger number of high confidence boxes from either the the appearance or the flow network. When a single action is present in each video, as in JHMDB, *boost-fusion* perform better (Table 4.2). In the Table 4.3, we present a complete

class-wise performance comparison of the two fusion strategies on both datasets.

Table 4.2: Action detection results (mAP) on J-HMDB-21.

IoU threshold δ	0.2	0.5	0.75	avg-mAP
Gkioxari and Malik [8]†	–	53.3	–	–
Wang *et al.* [132]†	–	56.4	–	–
Weinzaepfel *et al.* [13]†	63.1	60.7	–	–
Saha *et al.* [14]†	72.6	71.5	43.3	40.0
Peng and Schmid [90]†	**74.1**	**73.1**	–	–
Ours: Appearance (A)*	60.8	59.7	37.5	33.9
Ours: Real-time flow (RTF)*	56.9	47.4	20.2	19.3
Ours: A + RTF (union-set)*	66.0	63.9	35.1	34.4
Ours: A + RTF (boost-fusion)*	67.5	65.0	36.7	38.8
Ours: Accurate - flow (AF)**	68.5	67.0	38.7	36.1
Ours: A + AF (union-set)**	70.8	70.1	43.7	39.7
Ours: A + AF (boost-fusion)**	73.8	72.0	**44.5**	**41.6**
SSD+ [14] A + AF (boost-fusion)†	73.2	71.1	40.5	38.0

* Incremental & real-time ** Incremental, non real-time † Offline

Evaluation on J-HMDB-21. Table 4.2 reports action detection results averaged over the three splits of *J-HMDB-21*, and compares them with those to our closest (offline) competitors. Our framework outperforms the multi-stage approaches of [8, 13, 132] in non-real-time mode at the standard IoU threshold of 0.5, while it attains figures very close to those of [14, 90] (73.8 versus 74.1 and 72.6, respectively) approaches, which make use of a two-stage Faster-RCNN.

Once again it is very important to point out that [90] employs a battery of frame-level detectors, among which one based on strong priors on human body parts. Our approach does not make any prior assumption on the object(s)/actor(s) performing the action of interest, and is thus arguably more general-purpose.

4.7.4 Relative contribution of tube generation and SSD

As anticipated we evaluated the offline tube generation method of [14] using the detection bounding boxes produced by the SSD network, to both provide a fair comparison and to understand each component's influence on performance. The related results appear in the last row of Table 4.1 and Table 4.2. Similiar comparison is shown in Table 4.3, it shows that most of the classes benifit from proposed tube construction method 4.4.

From comparing the figures in the last two rows of both tables it is apparent that our online tube generation performs better than the offline tube generation of [14], especially providing significant improvements at higher detection thresholds for both datasets. We can infer that the increase in performance comes

from both the higher-quality detections generated by SSD, as well as our new online tube generation method. The fact that our tube generation is online, greedy and outperforms offline methods, so it suggests that offline approaches have a big room for improvements.

The reason for not observing a big boost due to the use of SSD on JHMDB maybe its relatively smaller size, which does not allow us to leverage on the expressive power of SSD models. Nevertheless, cross-validating the CNNs' hyper-parameters (e.g. learning rate), might lead to further improvements there as well.

4.7.5 Contribution of the flow stream

The optical flow stream (§ 4.5) is an essential part of the framework. Fusing the real-time flow stream with the appearance stream (A+RTF mode) on UCF101-24 leads to a 2.1% improvement at $\delta = 0.5$. Accurate flow adds a further 3.3%. A similar trend can be observed on JHMDB-21, where A+RTF gives a 5% boost at $\delta = 0.5$, and the A+RTF mode takes it further to 72%. It is clear from Table 4.1 and Table 4.2 that optical flow plays a much bigger role on the JHMDB dataset as compared to UCF101-24. Real-time OF does not provide as big a boost as accurate flow, but still pushes the overall performance towards that of the top competitors, with the invaluable addition of real-time speed.

We also report an ablation study of the online spatiotemporal action detection performance on UCF101-24 dataset. Table 4.3 shows the class-specific video AP (average precision in %) for each action category of UCF101-24 generated by the appearance (A) model, appearance plus real-time flow (A & RTF) fusion model and appearance plus accurate flow (A & AF) fusion model. Results are generated at a spatiotemporal overlap threshold of $\delta = 0.2$ in Table 4.3. For 12 out of 24 action classes, our *appearance plus accurate flow* fusion technique gives the best APs. The appearance-based detection network alone achieves the best APs for three classes: "HorseRiding", "Surfing" and "Fencing". It is worth noting that for action classes "HorseRiding" and "Fencing", appearance cues such as "horse" and "fencing player with white dress" are the most discriminative features. Whereas, in action class "Surfing", sea motion might be preventing the flow networks from learning discriminative features.

A difference between the two fusion strategies can be observed (Table 4.3) in classes with multiple actors, such as "IceDancing", "SalsaSpin" and "Biking". There, union-set fusion exhibits a significant improvement when compared with boost-fusion strategy.

Difficult classes. "Basketball", "CricketBowling", "VolleyballSpiking" and

Table 4.3: Spatiotemporal action detection results (video APs in %) on UCF101-24 at $\delta = 0.2$ along with class-wise statistics about UCF101-24 dataset in first two rows (number of action instance per video and action instance duration compared to video duration). All 24 claases are presented in following order: Basketball (1-BB), BasketballDunk (2-BD), Biking (3-BK), CliffDiving (4-CD), CricketBowling (5-CB), Diving (6-DV), Fencing (7-FC), FloorGymnastics (8-FG), GolfSwing (9-GS), HorseRiding (10-HR), IceDancing (11-ID), LongJump (12-LJ), PoleVault (13-PV), RopeClimbing (14-RC), SalsaSpin (15-SS), SkateBoarding (16-SB), Skiing (17-SK), Skijet (18-SI), SoccerJuggling (19-SJ), Surfing (20-SF), TennisSwing (21-TS), TrampolineJumping (22-TJ), VolleyballSpiking (23-VS), and WalkingWithDog (24-WD)

Actions	1-BB	2-BD	3-BK	4-CD	5-CB	6-DV	7-FC	8-FG
Number of Actions/Video	1.0	1.0	1.8	1.0	1.1	1.0	2.4	1.0
Action/Video duration (ratio)	0.34	0.59	0.70	0.64	0.36	0.65	0.89	1.00
Saha *et al.* [14]	39.6	49.7	66.9	73.2	14.1	93.6	85.9	99.8
Ours-Appearance (A)	43.0	67.4	75.8	67.2	50.5	100.0	88.5	97.9
Ours-A + RTF (boost-fusion)	42.2	69.0	71.7	73.1	41.3	100.0	87.6	99.1
Ours-A + RTF (union-set)	42.0	64.6	73.7	75.2	41.5	100.0	86.5	97.9
Ours-A + AF (boost-fusion)	45.0	86.4	67.6	78.2	44.2	100.0	89.8	99.9
Ours-A + AF (union-set)	43.9	81.6	73.6	73.7	49.0	100.0	90.2	97.9
SSD+ [14] A + AF (union-set)	43.2	78.5	65.8	72.0	43.6	100.0	86.1	98.1

Actions	9-GS	10-HR	11-ID	12-LJ	13-PV	14-RC	15-SS	16-SB
Number of Actions/Video	1.0	1.0	2.3	1.0	1.1	1.0	4.9	1.0
Action/Video duration (ratio)	1.00	0.95	0.86	0.53	0.30	0.65	0.31	0.84
Saha *et al.* [14]	68.3	94.1	63.1	57.2	75.1	89.6	31.1	85.1
Ours-Appearance (A)	59.9	95.9	56.5	59.7	80.8	93.4	36.9	86.1
Ours-A + RTF (boost-fusion)	64.3	96.0	72.8	68.8	72.7	94.5	19.7	85.6
Ours-A + RTF (union-set)	62.1	96.0	77.6	69.7	76.1	96.1	22.2	87.4
Ours-A + AF (boost-fusion)	65.8	96.0	74.0	81.4	80.3	95.8	23.1	88.3
Ours-A + AF (union-set)	62.0	96.0	76.3	82.9	82.7	98.1	25.7	87.8
SSD+ [14] A + AF (union-set)	61.3	96.0	60.6	83.4	84.6	98.6	20.1	88.4

Actions	17-SK	18-SI	19-SJ	20-SF	21-TS	22-TJ	23-VS	24-WD
Number of Actions/Video	1.0	1.0	1.1	1.6	1.3	2.5	1.1	1.1
Action/Video duration (ratio)	0.67	0.95	0.85	0.98	0.81	1.00	0.34	1.00
Saha *et al.* [14]	79.6	96.1	89.1	63.2	33.6	52.7	20.9	75.6
Ours-Appearance (A)	78.6	94.6	71.0	65.4	37.6	59.4	24.8	83.7
Ours-A + RTF (boost-fusion)	78.8	89.9	82.1	62.1	31.7	57.7	27.0	83.2
Ours-A + RTF (union-set)	81.0	87.1	82.4	62.1	37.4	59.4	21.7	85.1
Ours-A + AF (boost-fusion)	80.0	91.1	93.1	65.4	38.7	57.4	28.3	84.3
Ours-A + AF (union-set)	81.8	93.2	93.0	65.8	38.2	58.5	26.1	86.9
SSD+ [14] A + AF (union-set)	82.0	93.5	92.7	61.1	38.8	56.8	30.7	86.6

"TennisSwing" are the most difficult classes. Most of the "Basketball" training videos have at least one actor (basketball player) present in the video. The "Basketball" action, however, is performed within a small temporal extent. As temporal detection is difficult, action categories with a relatively large number of temporally untrimmed test videos such as "CricketBowling" , "Volleyball-Spiking" and "TennisSwing" show lower APs. Further, running (during the

"CricketBowling" action) is not considered as a part of the action which makes it even more difficult to detect. "VolleyballSpiking" videos contain many potential actors (Volleyball players) which are difficult to distinguish. It is clear from the above pieces of evidence that, it is necessary to re-train network using a background frame, which is not done. To achieve that we would need to modify the cost function of SSD to accept an image without any positive instance. At the moment it requires at least one positive instance present in any video frame.

Easy classes. "FloorGymnastics", "HorseRiding" and "SoccerJuggling" are the easiest classes to detect, possibly because these classes contain mostly one actor at a time and have salient appearance features. An example, consider the presence of a horse in the "HorseRiding" class.

Dataset statistics of UCF101-24 dataset. The first two rows of Table 4.3 shows class-wise statistics on the UCF101-24 dataset. The number of action instances per video is shown in the first row, averaged over test-list of split 1. The duration of action instance compared with the duration of the video is shown in the second row, averaged over test-list of split 1. It is clear that the most difficult classes are those which span a shorter fraction of the entire video duration.

4.7.6 Test time detection speed

To support our claim to real time capability, we report the test time detection speed of our pipeline under all three types of input A (RGB), A+RTF (real-time flow), A + AF (accurate flow) in Table 4.4. These figures were generated using a desktop computer with an Intel Xeon CPU@2.80GHz (8 cores) and two NVIDIA Titan X GPUs. Real-time capabilities can be achieved by either not using optical flow (using only appearance (A) stream on one GPU) or by computing real-time optical flow [23] on a CPU in parallel with two CNN forward passes on two GPUs. For action tube generation (§ 4.4) we ran 8 CPU threads in parallel for each class. We used the real-time optical flow algorithm [23] in a customised setting, with a minimum number of pyramid levels set to 2 instead of 3, and patch overlap 0.6 rather than 0.4. OF computation averages ~ 7 ms per image.

Table 4.4 also compares our detection speed to that reported by Saha *et al.* [14] (see Chapter 3). With an overall detection speed of 40 fps (when using RGB only) and 28 fps (when using also real-time OF), our framework is able to detect multiple co-occurring action instances in real-time while retaining very competitive performance.

Table 4.4: Test time detection speed.

Framework modules	A	A+RTF	A+AF	[14]
Flow computation (ms*)	–	7.0	110	110
Detection network time (ms*)	21.8	21.8	21.8	145
Tube generation time (ms*)	2.5	3.0	3.0	10.0
Overall speed (fps**)	40	28	7	4

* ms - milliseconds ** fps - frame per second.

4.7.7 Qualitative results

Fig. 4.5 and 4.6 show qualitative results of spatiotemporal action detection on temporally untrimmed "Fencing" and "Surfing" action sequences taken from UCF101-24 testset. Fig. 4.7 shows sample early action label prediction and online action detection qualitative results on the J-HMDB-21 dataset. Fig. 4.8 provides additional evidence on the action detection performance of our method on UCF101-24 dataset.

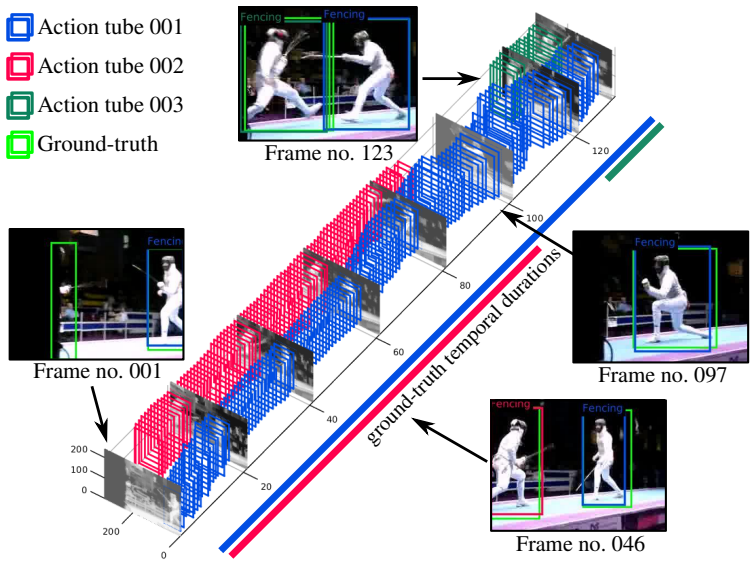


Figure 4.5: Sample spatiotemporal action detection results on a "Fencing" action sequence taken from UCF101-24 testset. The detected action tubes are plotted in 3D and drawn in different colour indicating 3 different action instances. The ground-truth temporal duration of each action instance is shown by the coloured bars. Note that the temporal duration of the detected and the ground-truth tubes are closely matched (with good temporal overlaps).

4.8 Summary and limitations

Summary: We presented a novel online framework for action detection and prediction able to address the challenges involved in concurrent multiple human action recognition, spatial detection and temporal detection, in real-time.

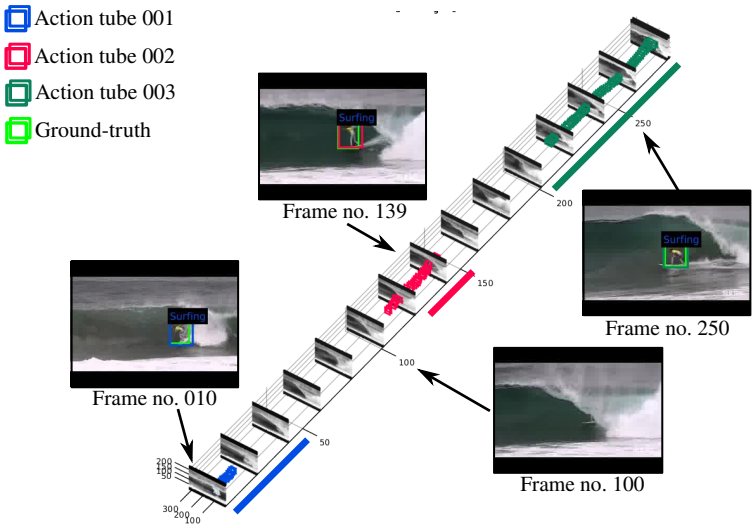


Figure 4.6: Sample spatiotemporal action detection results on a "Surfing" action sequence taken from UCF101-24 testset. The detected action tubes are plotted in 3D and drawn in different colour indicating 3 different action instances. The ground-truth temporal duration of each action instance is shown by the coloured bars. Note that the temporal durations of the detected and the ground-truth tubes are closely matched.

Thanks to an efficient deep learning strategy for the simultaneous detection and classification of region proposals and a new incremental action tube generation approach, our method achieves superior performances compared to the previous state-of-the-art on early action prediction and online detection, while outperforming the top offline competitors, in particular at high detection overlap. Its combination of high accuracy and fast detection speed at test time paves the way for its application to real-time applications such as autonomous driving, human-robot interaction and surgical robotics, among others.

Limitations: as for the previously illustrated offline action tubes detection method, the proposed online method retains the same limitations in terms of temporal reasoning, for the latter is limited to the tube construction method (§ 4.4) and the optical flow stream (§ 4.3.2).

Looking ahead: in the next chapter, we propose to solve these limitations by devising a closer-to-optimal formulation for the entire action tube detection problem. We propose to address tube detection by dividing it into a set of sub-problems, in which the detection of a 'micro-tube' from a small set of consecutive frames is one of the sub-problems. As shown in Chapter 5, the proposed solution also helps with the linking formulation.

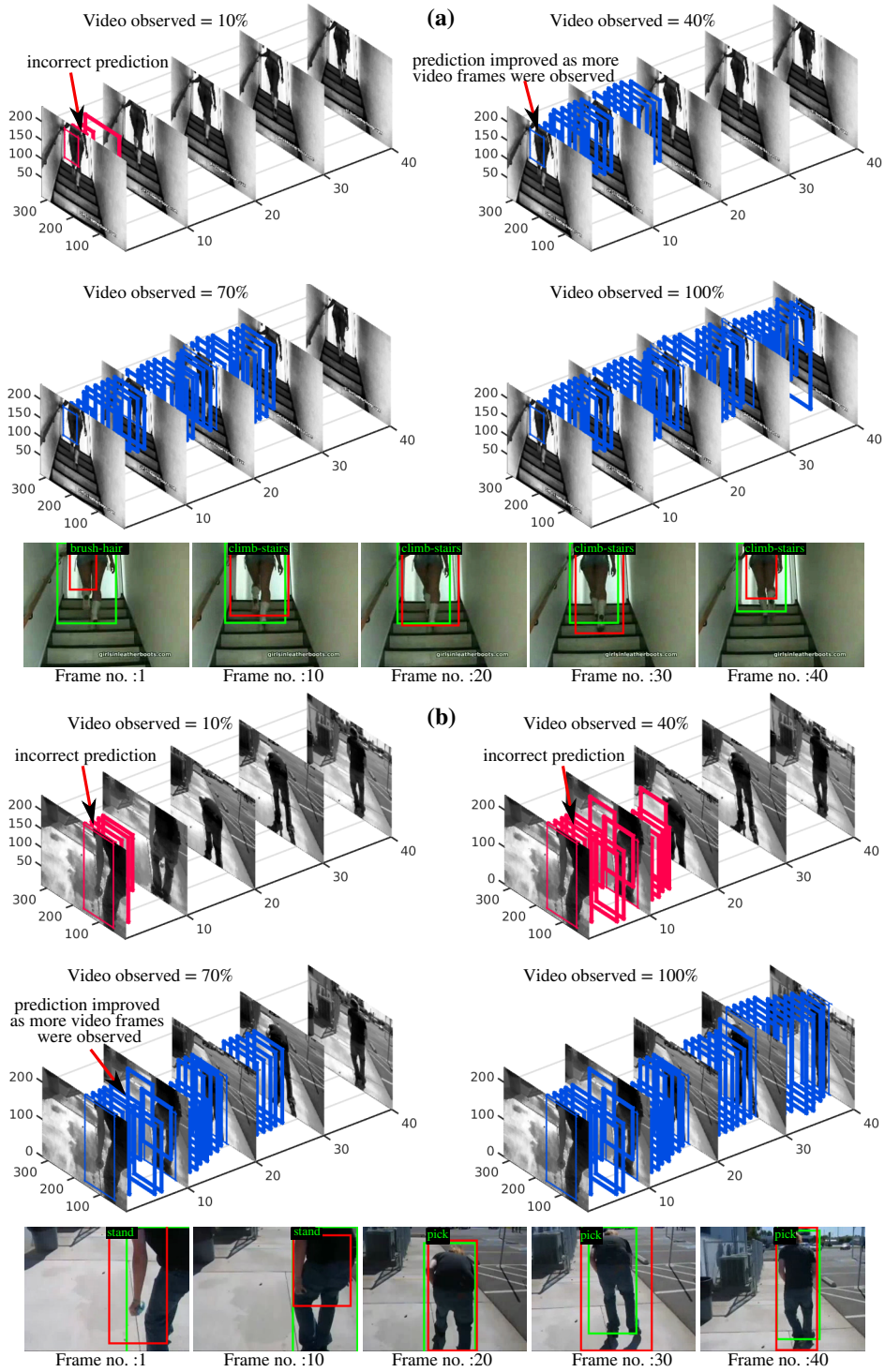


Figure 4.7: Sample early action label prediction and online action detection results on J-HMDB-21 dataset. ***(a)*** *and* ***(b)*** *show prediction results of 2 different test videos with ground-truth action labels 'climb-stairs' and 'pick' respectively. Each video and its corresponding space-time detection tube were plotted in 3D at different time points (i.e., % of video observed). Detection tubes are drawn in two different colours to indicate the wrong early label prediction and the improved prediction as more video frames were observed in time. Below the 3D plot, the predicted action labels for the same video at different time points are overlaid on the corresponding video frames in which the green box depicts the ground-truth and red depicts the predicted bounding box.*



Figure 4.8: Sample action detection results on UCF101-24. Each row represents a UCF101-24 test video clip. Ground-truth bounding boxes are drawn in green and detection boxes are in red.

Part II : Micro-tubes for Action Detection and Future Prediction of Tubes

Chapter 5

Transition Matrix Network for Flexible Micro-tube Proposal Generation

5.1 Introduction

A frame-level action detection approach, such as those presented in the previous two chapters, are sub-optimal for long term action tube detection, for they do not allow any feature-level interaction across frames.

In this chapter we take two steps towards a truly optimal solution of the action detection problem by considering video-level region proposals formed by a pair of bounding boxes spanning two successive video frames at an arbitrary temporal interval Δ. This allows us to learn a feature representation from pairs of video frames, unlike the single frame-based approaches of Chapter 3 and Chapter 4. It is quite easy to stack multiple frames and pass them through a CNN, but we also need to solve the feature-level correspondence problem, so that we can take features from both frames into account with respect to the reference anchor boxes. Unlike frame-based actor/object detectors, we need reference anchor boxes to span multiple frames.

5.1.1 AMTNet - 3D proposals for micro-tube detection

3D proposals can be created by extending 2D frame-level anchors/proposals in the temporal dimension. The networks will then learn a feature representation from pairs of video frames, allowing them to implicitly learn the temporal correspondence between inter-frame action regions (bounding boxes). As a result, they can predict action *micro-tubes* [24], i.e., temporally linked frame-level detections for short sub-sequences of a test video clip. We call the resulting

architecture an *action micro-tube network* (AMTNet) [24]. In the following we demonstrate that AMTNet does improve performance in comparison with a frame-based approach by a large margin.

A major concern with AMTNet is that action proposals are generated by extending 2D object proposals (anchor/prior boxes for images) [21, 81] to 3D proposals (anchor cuboids spanning multiple frames) (see Fig. 5.1(a)). These 3D anchor cuboids may be suitable for "static" actions (e.g. "handshake" or "clap", in which the spatial location of the actor(s) does not vary over time, see Figure 5.1(c)), but are most inappropriate for "dynamic" ones (e.g. "horse riding", "skiing"). Figure 5.1(a) underscores this issue. In the remainder of the chapter we thus illustrate a way to handle this problem. However, towards the end of this chapter, we show evidence that, even considering these issues, the anchor cuboid-based AMNet possesses multiple advantages, and improves on the single frame-based approaches presented in the previous chapters.

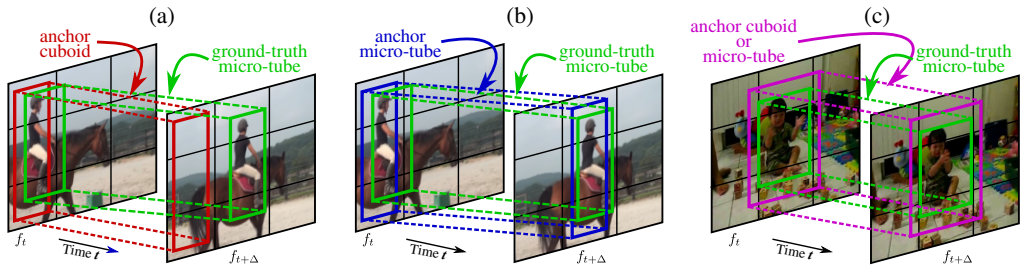


*Figure 5.1: Illustrating the key limitation of anchor cuboids using a "dynamic" action such as "horse riding". **(a)** A horse rider changes its location from frame f_t to $f_{t+\Delta}$ as shown by the ground truth bounding boxes (in green). As anchor cuboid generation [15, 24] is constrained by the spatial location of the anchor box in the first frame f_t, the overall spatiotemporal IoU overlap between the ground-truth micro-tube and the anchor cuboid is relatively low. **(b)** In contrast, anchor micro-tube proposal generator is much more flexible, as it efficiently explores the video search space via an approximate transition matrix estimated according to a hidden Markov model (HMM) formulation. As a result, the anchor micro-tube proposal (in blue) generated by the proposed model exhibits higher overlap with the ground-truth. **(c)** For "static" actions (such as "clap") in which the actor does not change location over time, anchor cuboid and anchor micro-tubes have the same*

5.1.2 TraMNet - Flexible 3D proposals

We observed in Figure 5.1 that anchor cuboids are limiting for dynamic actions. However, the video proposal search space ($\mathcal{O}(n^f)$) is much larger than the image proposal search space ($\mathcal{O}(n)$), where n is the number of anchor boxes per frame and f is the number of video frames considered.

As a next step, we thus propose a flexible micro-tube proposal generation method which allows us to explore the video search space efficiently via an

approximate transition matrix. Such transition matrix encodes the probability of a temporal link between an anchor box at time t and one at $t + \Delta$, and is estimated within the framework of discrete state/continuous observation hidden Markov models (HMMs, see Section 5.4) [145]. Here, the hidden states are the 2D bounding-box coordinates $[x_{min}, y_{min}, x_{max}, y_{max}]'$ of each anchor box from a (finite) hierarchy of fixed grids at different scales. Anchor micro-tubes are not bound to be of strictly cuboidal shape (as in [15, 16, 24]), thus resulting in a higher IoU overlap with the ground-truth (see the anchor micro-tubes in Figure 5.1(b)), specifically for instances where the spatial location of the actor changes significantly from f_t to $f_{t+\Delta}$ in a training pair. We thus propose a novel configurable deep neural network architecture (see Fig. 5.2), which leverages high-quality micro-tubes shaped by learnt anchor transition probabilities. We call it the *transition matrix network* (TraMNet).

In Section 5.4.3 we show that AMTNet is special case of TraMNet , i.e., a TraMNet can be converted into an AMTNet by replacing the learnt probabilistic transition matrix with the identity matrix.

To validate these notions we produced a new action detection dataset which is a "transformed" version of UCF101-24 [1], in which we force action instances to be dynamic (i.e., to change their spatial location significantly over time) by introducing random translations in the 2d spatial domain. We show that our proposed action detection approach outperforms the baseline AMTNet method when trained and tested on this transformed dataset.

5.1.3 Handling sparse annotations

The previous action detection methods such as as [15,16] require dense ground-truth annotation for network training: bounding-box annotation is required for k consecutive video frames, where k is the number of frames in a training example. Kalogeiton *et al.* [15] use $k = 6$ whereas for Hou *et al.* [16] $k = 8$. Generating such dense bounding box annotation for long video sequences is highly expensive and impractical [11, 125]. The latest generation action detection benchmarks DALY [125] and AVA [11], in contrast, provide sparse bounding-box annotations. More specifically, DALY has 1 to 5 frames bounding box annotation per action instance, irrespective of the duration of an instance, whereas AVA only has one frame annotation per second. This motivates the design of a deep network capable of handling sparse annotations, while still being able to predict micro-tubes over multiple frames.

Unlike [15,16], AMTNet uses *pairs* of successive frames $(f_t, f_{t+\Delta})$, thus eliminating the need for dense training annotation when Δ is large (e.g., $\Delta =$

$\{5, 10, 21\}$) or arbitrary as in DALY [125]. If the spatiotemporal IoU (Intersection over Union) overlap between ground-truth micro-tube and action proposal could be improved (cf. Fig. 5.1), such a network would be able to handle sparse annotation (e.g., pairs of frames which are $\Delta = 21$ apart). Indeed, the use of pairs of successive frames $(f_t, f_{t+\Delta})$, in combination with the flexible anchor proposals introduced here, is arguably more efficient than any other state-of-the-art method [3, 16, 24] in handling sparse annotations (e.g. DALY [125] and AVA [11]), which we quantitatively demonstrate later in the chapter.

Related publications: The AMNet work presented in this chapter appeared in ICCV 2017 [24]. The dissertation author was the main co-investigator in [25]. Here, we present multiple improvements over the original work [24], namely: i) we incorporate the SSD architecture to keep the pipeline real-time, as compared to the Faster R-CNN used in [24]; ii) we adopt the online tube construction method [9] of Chapter 4 because of its performance and its online processing capabilities. The TraMNet work presented in this chapter appeared in ACCV 2018 [25]. The dissertation author was the primary investigator in [25].

Outline: The remainder of the chapter is organised as follows. We start by presenting an overview of our TraMNet based action detection approach in Section 5.2. In Section 5.3, we introduce the base network architecture used for feature learning. We cast the action proposal generation problem in a hidden Markov model (HMM) formulation (§ Section 5.4), and introduce an approximate estimation of the HMM transition probability matrix using a heuristic approach (§ Section 5.4.1). In Section 5.5, a configurable pooling layer architecture is presented which pools convolutional features from the regions in the two frames linked by the estimated transition probabilities. Finally, the output layers of the network (i.e., the micro-tube regression and classification layers) are described in Section 5.5.2. Next, we report the experimental validation of the proposed model in Section 5.7 from both qualitative and quantitative perspective. Finally, we present a summary and comparison among AMTNet and TraMNet in Section 5.8.

5.2 Overview of the approach.

The network architecture of AMTNet and TraMNet builds on the architectural components of SSD [21] (Fig. 5.2). The proposed network takes as input a pair of successive video frames $f_t, f_{t+\Delta}$ (where Δ is the inter-frame distance) (Fig. 5.2 (a)) and propagates these frames through a base network comprised

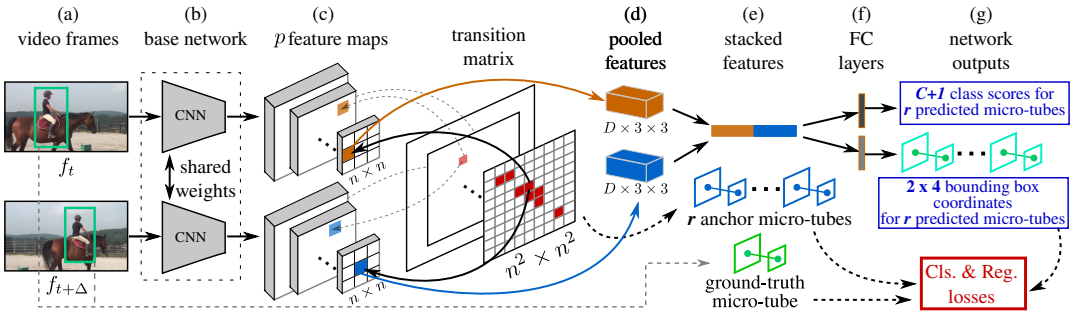


Figure 5.2: Overview of our proposed TraMNet at training time.

of two parallel CNN networks (§ 5.3 Fig. 5.2 (b)), which produce two sets of p conv feature maps K_p^t and $K_p^{t+\Delta}$ forming a pyramid (Fig. 5.2 (c)). These feature pyramids are used by a configurable pooling layer (§ 5.5 and Fig. 5.2 (d)) to pool features based on the transition probabilities defined by a transition matrix $\mathbf{A}$, probabilistic matrix for TraMNet (§ 5.4.1) and identity matrix for AMTNet (§ 5.4.3). The pooled convolutional features are then stacked (§ 5.5 and Fig. 5.2 (e)), and the resulting feature vector is passed to two parallel fully connected (linear) layers (one for classification and another for micro-tube regression, see § 5.5.2 and Fig. 5.2 (f)), which predict the output micro-tube and its classification scores for each class C.

Each training mini-batch is used to compute the classification and micro-tube regression losses given the output predictions, ground truth and anchor micro-tubes. We call our network *"configurable"* because the configuration of the pooling layer (see Fig. 5.2 (d)) depends on the transition matrices $\mathbf{A}$, and can be changed by altering the threshold applied to $\mathbf{A}$ (see Section 5.5.1) or by replacing the transition matrices with a new one for another dataset or having a different transition matrices at test time (see § 5.5.3).

5.2.1 Contributions

In summary, we present a novel deep learning architecture for spatiotemporal action detection which:

- on the methodological side, constitutes a key conceptual step forward from action detection paradigms relying on frame-level region proposals towards networks able to regress optimal solutions to the problem in the form of complete action tubes;

- a novel, end-to-end trainable deep network architecture which addresses the spatiotemporal action detection and classification task jointly using a single round of optimisation;

- introduces an efficient and flexible anchor micro-tube hypothesis generation framework to generate high-quality action proposals;
- can handle significant spatial movement in dynamic actions without penalising more static ones;
- is a scalable solution for training models on both sparse and dense annotations.

5.3 Base network

The base network takes as inputs a pair of video frames $(f_t, f_{t+\Delta})$ and propagates them through two parallel CNN streams (cf. Fig. 5.2 (b)). In Fig. 5.3 (a), we show the network diagram of one of the CNN streams; the other follows the same design.

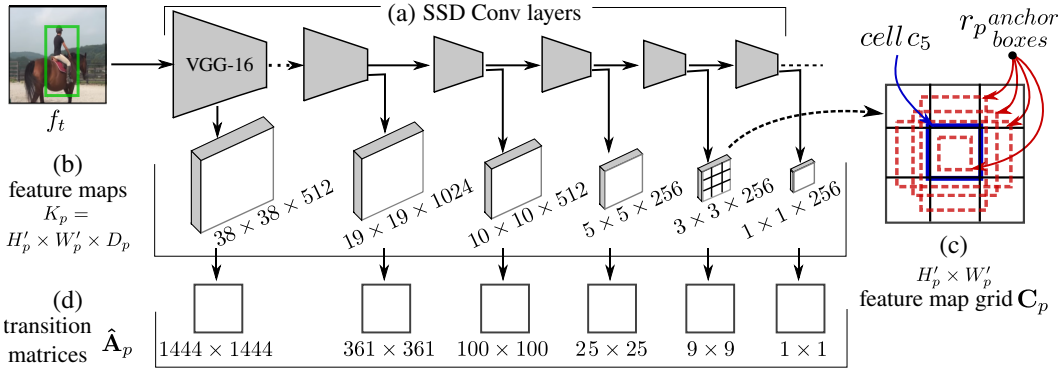


Figure 5.3: Base network architecture. ***(a)*** *SSD convolutional layers;* ***(b)*** *the corresponding convolutional feature maps outputted by each convolution layer;* ***(c)*** *r anchor boxes with different aspect ratios assigned to cell location c_5 of the 3×3 feature map grid;* ***(d)*** *transition matrices for the P feature map grids in the pyramid, where $P = 6$.*

The network architecture is based on that of the Single-Shot-Detector (SSD) [21]. The CNN stream outputs a set of P convolutional feature maps K_p, $p = \{1, 2, ..., P = 6\}$ (feature pyramid, cfr. Fig. 5.3 (b)) of shape $[H'_p \times W'_p \times D_p]$, where H'_p, W'_p and D_p are the height, width and depth of the feature map at network depth p, respectively. The feature maps are extracted at different depths $p = \{1, 2, ..., 6\}$ of the network. For $P = 6$, the convolutional feature map spatial dimensions are $H' = W' = \{38, 19, 10, 5, 3, 1\}$, respectively. The feature maps at the lower depth levels (i.e., $p = 1, 2$ or 3) are responsible for encoding smaller objects/actions, whereas feature maps at higher depth levels encode larger actions/objects. For each cell location c_{ij} of $[H'_p \times W'_p]$ feature map grid $\mathbf{C_p}$, r anchor boxes (each with different aspect ratio or scale) are assigned where $r_p = \{4, 6, 6, 6, 4, 4\}$. E.g. at each cell location of the 3×3 grid in

the pyramid, 4 anchor boxes are produced (Fig. 5.3 (c)), resulting in a total of $3 \times 3 \times 4 = 36$ anchor boxes. These anchor boxes, assigned for all $P = 6$ distinct feature map grids, are then used to generate action proposal hypotheses based on the transition probability matrix, as explained below.

Note that the proposed framework is not limited to any particular base network architecture, and is flexible enough to accommodate any latest 3D networks [17, 57, 146] or single-stage object detectors [146, 147].

5.4 HMM-based action proposal generation

A *hidden Markov model* (HMM) models a time series of (directly measurable) *observations* $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T\}$, either discrete or continuous, as randomly generated at each time instant t by a *hidden state* $\mathbf{q}_t \in \mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_N\}$, whose series form a Markov chain, i.e., the conditional probability of the state at time t given $\mathbf{q}_1, ..., \mathbf{q}_{t-1}$ only depends on the value of the state $\mathbf{q}_{t-1}$ at time $t-1$. The whole information on the time series' dynamics is thus contained in a *transition probability matrix* $\mathbf{A} = [p_{ij}; i, j = 1, .., n]$, where $p_{ij} = P(\mathbf{q}_j|\mathbf{q}_i)$ is the probability of moving from state i to state j, and $\sum_{j=1}^{N} p_{ij} = 1 \; \forall i$.
In our setting, a state $\mathbf{q}_n$ is a vector containing the 2D bounding-box coordinates of one of the anchor boxes $[x^a_{min}, y^a_{min}, x^a_{max}, y^a_{max}]'$ in one of the grids forming the pyramid (§ 5.3). The transition matrix encodes the probabilities of a temporal link existing between an anchor box (indexed by i) at time t and another anchor box (indexed by j) at time $t + \Delta$. The continuous observations $\mathbf{o}_t$, $t = 1, ..., T$ are the ground-truth bounding boxes, so that $\mathbf{O}$ corresponds to a ground-truth action tube.

In hidden Markov models, observations are assumed to be Gaussian distributed given a state $\mathbf{q}_i$, with mean $\mathbf{o}^i_\mu$ and covariance $\mathbf{Q}^i_\Sigma$. After assuming an appropriate distribution for the initial state, e.g. $P(\mathbf{q}_0) \sim \mathcal{N}(0, I)$, the transition model $A = [P(\mathbf{q}_j|\mathbf{q}_i)]$ allows us to predict at each time t the probability $P(\mathbf{q}_t|\mathbf{O}_{1:t})$ of the current state given the history of previous observations, i.e., the probability of each anchor box at time t given the observed (partial) ground-truth action tube. Given a training set, the optimal HMM parameters (A, $\mathbf{o}^i_\mu$ and $\mathbf{Q}^i_\Sigma$ for $i = 1, ..., N$) can be learned using standard expectation maximisation (EM) or the Baum-Welch algorithm, by optimising the likelihood of the predictions $P(\mathbf{q}_t|\mathbf{O}_{1:t})$ produced by the model.

Once training is done, at test time, the mean $\mathbf{o}^{\hat{\mathbf{q}}_t}_\mu$ of the conditional distribution of the observations given the state associated with the predicted state $\hat{\mathbf{q}}_t \doteq \arg\max_i P(\mathbf{q}_i|\mathbf{O}_{1:t})$ at time t can be used to initialise the anchor boxes for each of the P CNN feature map grids (§ 5.3). The learnt transition matrix

$\mathbf{A}$ can be used to generate a set of training action proposals hypotheses (i.e., anchor micro-tubes, Fig. 5.1). As in our case the mean vectors $\mathbf{o}_{\mu}^{i}$, $i = 1, ..., N$ are known a-priori (as the coordinates of the anchor boxes are predefined for each feature map grid, § 5.3), we do not allow the M-step of EM algorithm to update $\mathbf{Q}_{\mu} = [\mathbf{o}_{\mu}^{i}, i = 1, ..., N]$. Only the covariance matrix $\mathbf{Q}_{\Sigma}$ is updated.

5.4.1 Approximation of the HMM transition matrix

Although the above setting perfectly formalises the anchor box-ground truth detection relation over the time series of training frames, a number of computational issues arise. At training time, some states (anchor boxes) may not be associated with any of the observations (ground-truth boxes) in the E-step, leading to zero covariance for those states. Furthermore, for a large number of states (in our case $N = 8732$ anchor boxes), it takes around 4 days to complete a single HMM training iteration.

In response, we propose to approximate the HMM's transition probability matrix $\mathbf{A}$ with a matrix $\hat{\mathbf{A}}$ generated by a heuristic approach explained below. The problem is to learn a transition probability, i.e., the probability of a temporal link (edge) between two anchor boxes $\{b_t^a, b_{t+\Delta}^a\}$ belonging to two feature map grids $\mathbf{C}_p^t$ and $\mathbf{C}_{p'}^{t+\Delta}$. If we assume that transitions only take place between states at the same level $p = p'$ of the feature pyramid, the two sets of anchor boxes $\mathcal{B}_p^t = \{b_{t_1}^a, ..., b_{t_N}^a\}$ and $\mathcal{B}_p^{t+\Delta} = \{b_{(t+\Delta)_1}^a, ..., b_{(t+\Delta)_N}^a\}$ belonging to a pair of grids $\{\mathbf{C}_p^t, \mathbf{C}_p^{t+\Delta}\}$ are identical, namely: $\mathcal{B}_p^t = \mathcal{B}_p^{t+\Delta} \doteq \mathcal{B}_p = \{b_i^a, i = 1, ..., N\}$, allowing us to remove the time superscript. Recall that each feature map grid C_p has spatial dimension $[H'_p \times W'_p]$.

We compute a transition probability matrix $\hat{\mathbf{A}}_p$ *individually for each grid level* p, resulting in p such matrices of shape $[(H'_p)^2 \times (W'_p)^2]$ (see Fig. 5.3 (d)). For example, at level $p = 5$ we have a 3×3 feature map grids, so that the transition matrix $\hat{\mathbf{A}}_p$ will be $[3^2 \times 3^2]$. Each cell in the grid is assigned to r_p anchor boxes, resulting in $n = H'_p \times W'_p \times r_p$ total anchor boxes per grid (§ 5.3). Note that here we are interested in learning a transition probability between two anchor boxes belonging to two different cell locations in the grid, thus further reducing the search space complexity from $\mathcal{O}(n^f)$ to $\mathcal{O}(L^f)$ where, L is the number of cell locations in a grid.

5.4.2 Transition matrix computation for TraMNet

Initially, all entries of the transition matrix are set to zero: $\hat{\mathbf{A}}[i, j] = 0$. Given a ground-truth micro-tube $\mathbf{m}^g = \{b_t^g, b_{t+\Delta}^g\}$ (a pair of temporally linked ground-truth boxes [24]), we compute the IoU overlap for each ground-truth box with all

the anchor boxes $\mathcal{B}_p$ in the considered grid, namely: $IoU(b_t^g, \mathcal{B}_p)$ and $IoU(b_{t+\Delta}^g, \mathcal{B}_p)$. We select the pair of anchor boxes $\mathbf{m}^a = \{b_i^a, b_j^a\}$ (which we term *anchor micro-tube*) having the maximum IoU overlap with $\mathbf{m}^g$, where i and j are two cell locations. If $i = j$ (the resulting anchor boxes are in the same location) we get an anchor cuboid, otherwise a general anchor micro-tube.

It is repeated for all P feature map grids $\mathbf{C}_p$ to select the anchor micro-tube $\mathbf{m}_p^a$ with the highest overlap. The best match anchor micro-tube $\mathbf{m}_{\hat{p}}^a$ for a given ground-truth micro-tube $\mathbf{m}^g$ is selected among those P, and the transition matrix is updated as follows: $\hat{\mathbf{A}}[i,j] = \hat{\mathbf{A}}[i,j] + 1$. The above steps are repeated for all the ground-truth micro-tubes in a training set. Finally, each row of the transition matrix $\hat{\mathbf{A}}$ is normalised by dividing each entry by the sum of that row.

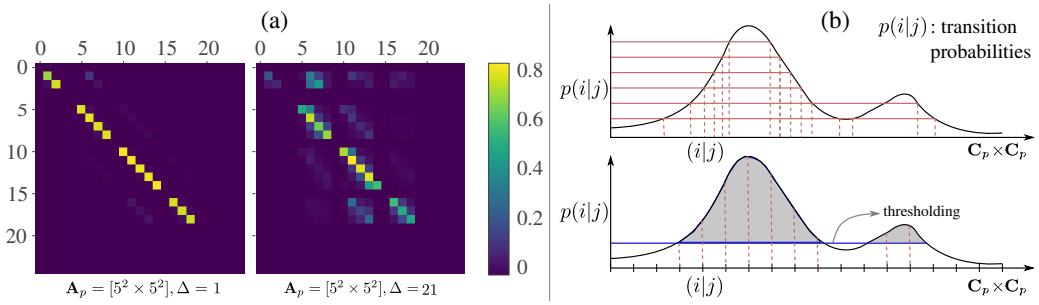


Figure 5.4: ***(a)*** *Transition matrix for a* 5×5 *feature map grid* ($p = 4$) *for different* Δ *values. As* Δ *increases, off-diagonal probability values also increase, indicating a need for anchor micro-tubes rather than anchor-cuboids.* ***(b)*** *Top - Monte Carlo sampling of transition hypotheses* $(i, j) \in \mathbf{C}_p \times \mathbf{C}_p$ *based on uniformly sampling the* $[0, 1]$ *range. Bottom - our anchor micro-tube sampling scheme, based on thresholding the transition probabilities* $p(i|j)$*, is also stochastic in nature and emulates Monte Carlo sampling. The blue line denotes the threshold and the shaded area above the threshold line shows the sampling region, a subset of the product grid* $\mathbf{C}_p \times \mathbf{C}_p$*.*

Fig. 5.4 plots the transition matrix $\hat{\mathbf{A}}_p$ for $p = 4$ (a feature map grid 5×5), for different values of Δ. As explained in the following, the configurable pooling layer employs these matrices to pool conv features for action proposal classification and regression.

Although our approach learns transition probabilities for anchor boxes belonging to the same feature map grid $\mathbf{C}_p$, we realise that the quality of the resulting action proposals could be further improved by learning transitions between anchors across different levels of the pyramid. As the feature dimension of each map varies in SSD, e.g. 1024 for $p = 2$ and 512 for $p = 1$, a more consistent network such as FPN [146] with Resnet [148] would be a better choice as base architecture. Here we stick to SSD to produce a fair comparison with [9, 15, 24], and leave this extension to future work.

5.4.3 Identity matrix for AMTNet

As explained in the introduction and overview, the AMTNet is a sub-case of TraMNet. We can implement AMTNet by replacing the transition matrix in TraMNet by an identity, i.e.

$$\hat{\mathbf{A}}[i,j] = \begin{cases} 1 & \text{if } i == j \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

In all of our experiment on AMTNet [24], we use similar implementation and hyperparameters as for TraMNet.

5.5 Configurable pooling layer

The SSD [21] network uses convolutional kernels of dimension $[3 \times 3 \times D]$ as classification and regression layers (called *classification* and *regression heads*). More specifically, SSD uses $r \times 4$ kernels for bounding box regression (recall r anchor boxes with different aspect ratios are assigned to each cell location (§ 5.3)) and $(C+1) \times r$ kernels for classification over the p convolutional feature maps (§ 5.3). This is fine when the number of proposal hypotheses is fixed (e.g., for object detection in images, the number of anchor boxes is set to 8732). In our setting, however, the number of proposals varies depending upon the cardinality $|\hat{\mathbf{A}}_p|$ of the transition matrix (§ 5.4.1). Consequently, it is more principled to implement the classification and regression heads as fully connected layers (see Fig. 5.2 (f)). If we observe consistent off-diagonal entries in the transition matrices (e.g. lots of cells moving one step in the same direction), we could perform pooling as convolution feature map stacking with padding to allow spatial movement. However, transition matrices are empirically extremely sparse (e.g., there are only 25 and 1908 off-diagonal non-zero entries in the transition matrices at Δ equal to 4 and 20, respectively, on the UCF101-24 dataset).

5.5.1 Anchor micro-tube sampling

Each transition matrix is converted into a binary one by a threshold so that the cardinality of the matrix depends not only on the data but also on the transition probability threshold. Our transition matrix-based anchor micro-tube sampling scheme is stochastic in nature and emulates Monte Carlo sampling techniques (Fig. 5.4 (b)). A threshold on the transition matrix allows us to sample a variable number of anchors rather than a fixed one. We empirically found that a 10% threshold gives the best results in all of our tests. We discuss the threshold

and its effect on performance in § 5.4.1.

The pooling layer (see Fig. 5.2 (d)) is configured to pool features from a pair of convolutional feature maps $\{K_p^t,\ K_p^{t+\Delta}\}$, each of size $[H'_p \times W'_p \times D]$. The pooling is done at cell locations i and j, specified by the estimated (thresholded) transition matrix $\hat{\mathbf{A}}_p$ (§ 5.4.1). The pooling kernel has dimension $[3 \times 3 \times D]$. Pooled features are subsequently stacked (see Fig. 5.2 (e)) to get a single feature representation of size $[2 \times 3 \times 3 \times D]$ per anchor micro-tube.

5.5.2 Classification and regression layers

After pooling and stacking, we get M conv features of size $[2 \times 3 \times 3 \times D]$, for each M anchor micro-tube cell regions, where $M = \sum_{p=1}^{P=6} |\hat{\mathbf{A}}_p|$ is the sum of the cardinalities of the P transition matrices. We pass these M features to a classification layer $((18 \times D), ((C+1) \times r))$, and a regression layer $((18 \times D), ((2 \times 4) \times r))$ (see Fig. 5.2 (f)). The classification layer outputs $C+1$ class scores and the regression layer outputs 2×4 bounding-box coordinates for r anchor micro-tubes per anchor micro-tube cell region (see Fig. 5.2 (g)). The linear classification and regression layers have the the same number of parameters as the convolutional heads in the SSD network [21].

5.5.3 Generality at test time

Our approach allows the pooling layer to be *"configurable"* for any transition matrix, while keeping the network parameters intact. This configurable nature provides us with the advantage of using different transition matrices at test time, we can generate a more generic set of anchor micro-tubes by setting the transition probabilities to 1 for those diagonal or off-diagonal entries which were earlier 0 for the training set, or accommodating all anchor micro-tubes transitions relative to each cell. At test time, in cases in which test Δ is larger than train Δ, we can use the Markov chain rule to generate the new transition matrix.

5.6 Online action tube generation

The output of the proposed network is a set of detection micro-tubes and their class confidence scores (see Fig. 5.2 (g)). Based on the previous chapter, we adapt the online action tube generation algorithm proposed by Singh *et al.* [9] (presented in Section 4.4.2 of Chapter 4) to compose these detection micro-tubes into complete action paths (tracklets) spanning the entire video. Note that,

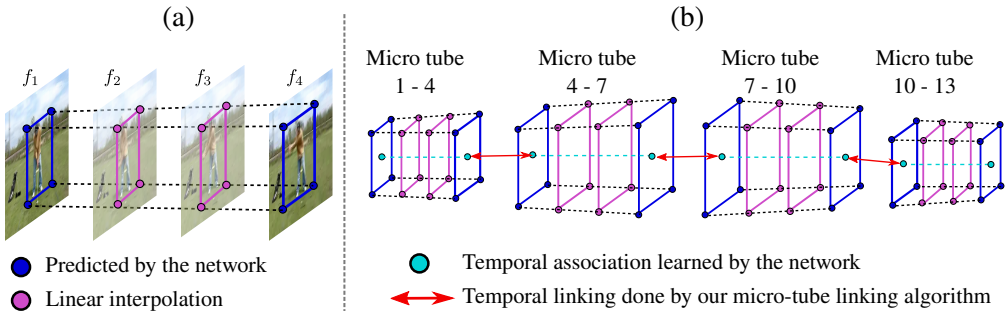


Figure 5.5: ***(a)*** *Action micro-tube generation by linear interpolation of coordinates of predicted bounding boxes. The blue circles denote the* 8 (x, y) *coordinate values of the bounding boxes predicted by our network for a pair of successive (but not necessarily consecutive) test video frames. The first frame in the pair is indexed by* $t = 1$*, the second by* $t = 4$*. We generate the coordinates of the detection bounding boxes (pink circles) for intermediate frames by linear interpolation of the predicted coordinates.* ***(b)*** *Action tube generation takes place via the linking of micro-tubes generated as in* ***(a)****. Note that, for a video sequence with* T *frames, our model only needs to perform* T/Δ *forward passes – as a result, the micro-tube linking algorithm (§ 5.6) only needs to connect* $T/\Delta - 1$ *of these frames.*

Singh *et al.* [9] use their tube generation algorithm to temporally connect frame-level detection bounding-boxes, whereas our modified version of the algorithm connects video-level detection micro-tubes. Similarly to [9], we build action paths incrementally by connecting micro-tubes across time.

Let B_t be the set of detection bounding boxes from a frame f_t. Since their method is based on a frame-level detector, Singh *et al.* [9] associate boxes in B_t to boxes in B_{t+1}. In our case, as TraMNet/AMTnet generates micro-tubes $m_t \in M_t \doteq B_t^1 \times B_{t+\Delta}^2$ each spanning a pair of frames $\{f_t, f_{t+\Delta}\}$ separated by an interval of Δ, the algorithm needs to link the micro-tubes associated with the pair $t, t+\Delta$ with those $m_{t+\Delta} \in M_{t+\Delta} \doteq B_{t+\Delta}^1 \times B_{t+2\Delta}^2$ associated with the following pair of frames $\{f_{t+\Delta}, f_{t+2\Delta}\}$. Note that the set of detections $B_{t+\Delta}^2$ generated for time $t+\Delta$ by the network when processing the first pair of frames, will in general differ from the set of detections $B_{t+\Delta}^1$ for the *same* frame $t+\Delta$ generated by the network when processing $\{f_{t+\Delta}, f_{t+2\Delta}\}$ frames.

Micro-tube linking happens by associating elements of $B_{t+\Delta}^2$, coming from M_t, with elements of $B_{t+\Delta}^1$, coming from $M_{t+\Delta}$, as shown in Figure 5.5(b). Interestingly, the latter is a relatively easier sub-problem, as all such detections are generated based on the same frame, unlike the across-frame association problem considered in [9]. Association is achieved based on both Intersection over Union (IoU) and class score, as the tubes are built separately for each class in a multi-label scenario. For more details, we refer the reader to [9].

5.6.1 Bounding box interpolation

At test time, for a input pair of successive video frames (f_t and $f_{t+\Delta}$), the final outputs of AMTnet (§ 5.2) are the top k detection micro-tubes and their class confidence scores. For example, in Figure 5.5 (a) an action micro-tube predicted by the network for an input pair (f_1, f_4) ($\Delta = 3$), composed by the two blue bounding boxes highlighted and uniquely determined by 8 coordinate values (blue circles) is depicted. In practice, the network predicts M such micro-tubes for each input pair, where M is a parameter which depends on the number of anchor locations. The predicted micro-tube, however, does not provide bounding boxes for the intermediate frames (in the example f_2 and f_3). We thus generate detection boxes for all intermediate frames using a simple coordinate wise linear interpolation.

5.6.2 Fusion of appearance and motion cues

We follow a late fusion strategy [15] to fuse appearance and motion cues, performed at test time after all the detections are extracted from the two streams. Kalogeiton *et al.* [15] demonstrated that *mean* fusion works better than both *boost* fusion [14] and *union-set* fusion [9]. Thus, in this chapter we produce all results using *mean* fusion [15]. We also report an ablation study of the appearance and flow stream performance in experiment Section (§ 5.7).

5.7 Experiments

We first present datasets, evaluation setup, fair comparison and implementation details used in Section 5.7.1. Here, we show how AMTNet and TraMNet is able to improve spatial-temporal action detection in Section 5.7.3. Thirdly, in Section 5.7.6, we discuss how a network learned using transition matrices is able to generalise at test time, when more general anchor-micro-tubes are used to evaluate the network. Finally, in Section 5.7.7, we quantitatively demonstrate that TraMNet can effectively handle sparse annotation as in the DALY dataset, and generalise well on the various train and test Δ's on the UCF101-24 dataset.

5.7.1 Datasets and evaluation setup

Similar to previous chapter and previous works [9, 15]. we evaluate our models on the UCF101-24 [1] to validate spatiotemporal detection performance. We use DALY [125] and UCF101-24 [1] to evaluate the method on sparse annotations.

Transformed-UCF101-24 was created by us by padding all images along both the horizontal and the vertical dimension. We set the maximum padding values to 32 and 20 pixels, respectively, as 40% of the average width (80) and height (52) of bounding box annotations. A uniformly sampled random fraction of 32 pixels is padded on the left edge of the image, the remaining is padded on the right edge of the image. Similar random padding is performed at the top and bottom of each frame. The padding itself is obtained by mirroring the adjacent portion of the image through the edge. The same offset is applied to the bounding box annotations.

Evaluation setup. Similar to previous chapter, we use *"average detection performance"* (avg-mAP) to compare AMTNet's and TraMNet's performance with the previous state-of-the-art approaches [9, 15, 90, 122] on UCF101-24 dataset.

On the DALY dataset, we also evaluate at various thresholds in both an untrimmed and a trimmed setting. The latter is achieved by trimming the action paths generated by the boundaries of the ground truth [125]. We further report the video classification accuracy using the predicted tubes as in [9], in which videos are assigned the label of the highest-scoring tube. One could improve classification performance on DALY by taking into consideration other tube scores. Nevertheless, in our tests, we adopt the existing protocol.

For ***fair comparison*** we re-implemented the methods of our previous[1] competitors [9,15] with SSD as the base network. As in our AMTNet/TraMNet networks, we also replaced SSD's convolutional heads with new linear layers. The same tube generation [9] and data augmentation [21] methods were adopted, and the same hyperparameters were used for training all the networks, including TraMNet and AMTNet. The only difference is that the anchor micro-tubes used in [15] were cuboidal for six frames, whereas AMTNet's and TraMNet's anchor micro-tubes are generated using transition matrices for two consecutive frames separated by Δ. We refer to these approaches as SSD-L (SSD-linear-heads) [9], and as ACT-L (ACT-detector-linear-heads) [15].

5.7.2 Network training and implementation details.

We used well-established training settings from previous chapters for all the above methods. While training on the UCF101-24 dataset, we used a batch size of 16 and an initial learning rate of 0.0005, with the learning rate dropping after $100K$ iterations for the appearance stream and $140K$ for the flow stream.

[1]This work was published in ACCV 2018, we compare with state-of-the-art approaches at the time of submission.

Whereas the appearance stream is only trained for $180K$ iterations, the flow stream is trained for $200K$ iterations. In all cases, the input image size was $3 \times 300 \times 300$ for the appearance stream, while a stack of five optical flow images [22] ($15 \times 300 \times 300$) was used as input for flow-stream. Each network was trained on 2 1080Ti GPUs. SSD-L and ACT-L are also trained using the same training setup.

Table 5.1: Action detection results on untrimmed videos from UCF101-24 split1. The table is divided into 5 parts. The first part lists approaches which have single frames as input; the second part approaches which take multiple frames as input; the third part contemplates the re-implemented versions of previous approaches [9, 15]; the fourth part reports the performance of AMTNet; lastly, we report TraMNet's performance.

Methods	Train Δ	Test Δ	$\delta = 0.2$	$\delta = 0.5$	$\delta = 0.75$	$\delta = .5{:}.95$	Acc %
T-CNN [16]	NA	NA	47.1	–	–	–	–
MR-TS [90]	NA	NA	73.5	32.1	02.7	07.3	–
Saha *et al.* [14]	NA	NA	66.6	36.4	07.9	14.4	–
SSD [9]	NA	NA	73.2	46.3	15.0	20.4	–
AMTnet [24] rgb-only	1,2,3	1	63.0	33.1	00.5	10.7	–
ACT [15]	1	1	76.2	49.2	19.7	23.4	–
RTPR (VGG) [95]	1	1	76.3	–	–	–	–
Gu *et al.* [11] ([90] + [17])	NA	NA	–	**59.9**	–	–	–
SSD-L with-trimming	NA	NA	76.2	45.5	16.4	20.6	92.0
SSD-L	NA	NA	76.8	48.2	17.0	21.7	92.1
ACT-L	1	1	77.9	50.8	19.8	**23.9**	91.4
AMTnet (ours)	1	1	**79.4**	**51.2**	19.0	23.4	**92.9**
AMTnet (ours)	5	5	77.5	49.5	17.3	22.5	91.6
AMTnet (ours)	21	5	76.2	47.6	16.5	21.6	90.0
TraMNet (ours)	1	1	79.0	50.9	**20.1**	**23.9**	92.4
TraMNet (ours)	5	5	77.6	49.7	18.4	22.8	91.3
TraMNet (ours)	21	5	75.2	47.8	17.4	22.3	90.7

5.7.3 Action detection performance

Table 5.1 shows the resulting performance on UCF101-24 at multiple train and test Δs for TraMNet versus other competitors [9, 14–16, 90]. Note that Gu *et al.* [11] build upon MS-TS [90] by adding a strong I3D [17] base network, making it unfair to compare [11] to SSD-L, AMTnet, ACT-L and TraMNet, which all use 2D VGG [96] as a base network.

The ACT is a dense network (processes 6 consecutive frames), which shows the best performance at high overlap (an avg-mAP of 23.9%). AMTnet is slightly inferior (23.4%), most likely due to its learning representations from only a pair of consecutive frames at its best training and test settings ($\Delta = 1$). TraMNet is able to match ACT-L's performance at high overlap (23.9%) while being comparatively more efficient (only using a pair of frames as compared to 6 used by ACT-L).

We cross-validated different transition probability thresholds on transition matrices. Thresholds of 2%, 5%, 10%, 15% and 20% yielded an avg-mAP of 21.6%, 22.0%, 22.4%, 21.9% and 21.2%, respectively, on the appearance stream. Given such evidence, we concluded that a 10% transition probability threshold was to be adopted throughout all our experiments.

5.7.4 AMTNet comparison with 2D methods

We SSD as base network for AMTNet and TraMNet, it is only fair to compare AMTNet's performance to that of SSD-L in Table 5.1. We can see that AMTNet performs better that the frame-level SSD-L method presented in the previous chapter.

5.7.5 TraMNet on Transformed-UCF101-24

The evaluation of AMTNet on Transformed-UCF101-24 (§ 5.7.1) shows an avg-mAP of 19.3% using the appearance stream only, whereas TraMNet records an avg-mAP of 20.5%, a gain of 1.2% that can be attributed to its estimating grid location transition probabilities. It shows that TraMNet is more suited to action instances involving substantial shifts from one frame to the next. A similar phenomenon can be observed on the standard UCF101-24 when the train or test Δ is greater than 1 in Table 5.1.

5.7.6 Location invariance at test time

Anchor micro-tubes are sampled based on the transition probabilities from specific cells (at frame f_t) to other specific cells (at frame $f_{t+\Delta}$) (§ 5.4.2) based on the training data. As a result, anchor miro-tubes are location dependent. However, as at test time action instances of the same class may appear in other regions of the image plane than those observed at training time, it is desirable to generate additional anchor micro-tubes proposals than those produced by the learnt transition matrices. Such *location invariance* property can be achieved at test time by augmenting the binary transition matrix (§ 5.5.1) with likely transitions from other grid locations.

Each row/column of the transition matrix $\hat{\mathbf{A}}$ (§ 5.4.2) corresponds to a cell location in the grid. One augmentation technique is to set all the diagonal entries to 1 (i.e., $\hat{\mathbf{A}}[i,j] = 1$, where $i == j$). This amounts to generating anchor cuboids which may have been missing at training time (cfr. Fig. 5.4 (a)). The network can then be evaluated using this new set of anchor micro-tubes by configuring the pooling layer (§ 5.5)) accordingly. When doing so, however,

Table 5.2: Action detection results (video-mAP) on the DALY dataset. SSD-L without trimming refers to when action paths are not trimmed and the network is SSD.

Methods	Test Δ	Untrimmed Videos			Trimmed Videos			
		δ=0.2	δ=0.5	Acc%	δ=0.5	δ=.5:.95	Acc%	CleaningFloor
Weinzaepfel *et al.* [125]	NA	13.9	–	–	63.9	–	–	–
SSD-L without-trimming	NA	06.1	01.1	61.5				
SSD-L	NA	**14.6**	**05.7**	58.5	63.9	38.2	75.5	80.2
AMTnet (ours)	3	12.1	04.3	62.0	63.7	39.3	76.5	83.4
TraMNet (ours)	3	13.4	04.6	**67.0**	**64.2**	**41.4**	**78.5**	**86.6**

we observed only a very minor difference in avg-mAP at the second decimal point for TraMNet. Secondly, we also evaluated TraMNet by incorporating the transitions from each cell to its 8 *neighbouring* cells (also at test time), but observed no significant change in avg-mAP.

A third approach, given a pyramid level p, and the initial binary transition matrix for that level, consists of computing the relative transition offsets for all grid cells (offset $= i - j \ \forall i, j$ where $\hat{\mathbf{A}}[i, j] = 1$). All such transition offsets correspond to different spatial translation patterns (of action instances) present in the dataset at different locations in the given video. Augmenting all the rows with these spatial translation patterns, by taking each diagonal entry in the transition matrix as a reference point, yields a more dense transition matrix whose anchor micro-tubes are translation invariant, i.e., spatial location invariant. However, after training TraMNet at train $\Delta = 1$ we observed that the final avg-mAP at test $\Delta = 1$ was 22.6% as compared to 23.9% when using the original (sparse) transition matrix. As in the experiments (i.e., added diagonal and neighbour transitions) explained above, we evaluated the network that was trained on the original transition matrices at train $\Delta = 1$ by using the transition matrix generated via relative offsets, observing an avg-mAP consistent (i.e., 23.9%) with the original results.

This shows that the system should be trained using the original transition matrices learned from the data, whereas more anchor micro-tube proposals can be assessed at test time without loss of generality. Hence, we can train our network efficiently and evaluate with more anchors as the evaluation is much faster than training. It also shows that UCF101-24 is not sufficiently realistic a dataset from the point of view of translation invariance, which is why we conducted tests on Transformed-UCF101-24 (§ 5.7.1) to highlight this issue.

5.7.7 Handling sparse annotations

Table 5.2 shows the results on the DALY dataset. We can see that TraMNet significantly improves on AMTnet and AMTNet on SSD-L in the trimmed video setting, with an average video-mAP of 41.4%. TraMNet reaches top classifica-

tion accuracy in both the trimmed and the untrimmed cases. As we would expect, TraMNet improves the temporal linking via better micro-tubes and classification, as clearly indicated in the trimmed videos setting. Nevertheless, SSD-L is the best when it comes to temporal trimming. We think this is because of each micro-tube in our case is 4 frames long as the test Δ is equal to 3, and each micro-tube only has one score vector rather than 4 score vectors for each frame, which might render temporal segmentation ineffective (loss of discriminative power across frames). Given the above evidence and in Section 5.7.4, we can say that AMTNet and TraMNet are better that spatial detection than temporal trimming as compared to single-frame approach defined by SSD-L from the previous chapter.

DALY allows us to show how TraMNet is able to handle sparse annotations better than AMTNet, which uses anchor cuboids, strengthening the argument that learning transition matrices help generate better micro-tubes.

TramNet's performance on 'CleaningFloor' at δ equal to 0.5 in the trimmed case highlights the effectiveness of general anchor micro-tubes for **dynamic classes**. 'CleaningFloor' is one of DALY's classes in which the actor moves spatially while the camera is mostly static. To further strengthen the argument, we picked classes showing fast spatial movements across frames in the UCF101-24 dataset and observed the class-wise average-precision (AP) at δ equal to 0.2. For 'BasketballDunk', 'Skiing' and 'VolleyballSpiking' TraMNet performs significantly better than both AMTnet and ACT-L; e.g. on 'Skiing', the performance of TraMNet, AMTNet and ACT-L is 85.2, 82.4 and 81.1, respectively.

5.7.8 Training and testing at multiple Δ's

To further test whether TraMNet can handle sparse annotation we introduced an artificial gap (Δ) in UCF101-24's training examples, while testing on frames that are far away (e.g. $\Delta = 30$). We can observe in Figure 5.6(b) that performance is preserved when increasing the training Δ while keeping the test Δ small (e.g. equal to 5, as shown in plot (a)). One could think of increasing Δ at test time to improve run-time efficiency: we can observe from Figure 5.6(a) that performance drops linearly as speed linearly increases. In both cases TraMNet consistently outperforms AMTNet. When Δ is large TraMNet's improvement is large as well.

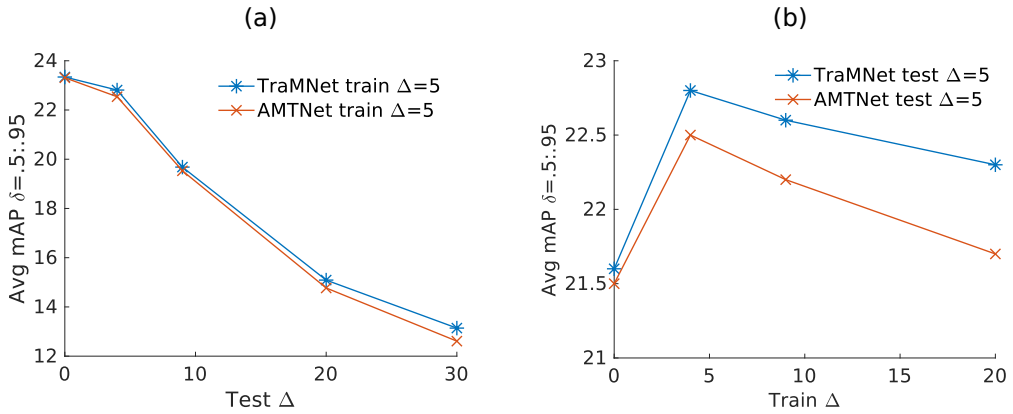


Figure 5.6: Avg mAP ($\delta = 0.5 : 0.95$) performance of TraMNet and AMTNet, (a) when tested at increasing Δ from 1 to 30 and trained at constant Δ equal to 5, (b) when tested at constant Δ equal to 5 and trained on increasing Δ from 1 to 20.

5.7.9 Temporal labelling

Temporal labelling is performed using the labelling formulation presented in [9] (similar to Chapter 3 and Chapter 4). Actually, temporal labelling hurts the performance on UCF101-24, as shown in Table 5.1 where 'SSD-L-with-trimming' uses [9]'s temporal labelling technique, whereas 'SSD-L' and the other methods below that do not. In contrast, on DALY the results are quite the opposite: the same temporal labelling framework improves the performance from 6.1% to 14.9% at $\delta = 0.2$. We think that these (superficially) contradictory results relate to the fact that action instances cover on average a very different fraction (70% versus 4%) of the video duration in UCF101-24 and DALY, respectively. Hence, we can conclude that temporal trimming does not have major impact on overall performance on UCF101-24 (§ 5.7.3) dataset except few difficult classes pointed out in Section 4.7.5 of Chapter 4.

5.7.10 Detection speed

Similar to previous chapters, we continue to keep our models real-time and online. We measured the average time taken for a forward pass for a batch size of 1 as compared to 8 by [9]. A single-stream forward pass takes 29.8 milliseconds (i.e. 33fps) on a single 1080Ti GPU. One can improve speed even further by evaluating TraMNet with Δ equal to 2 or 4, obtaining a 2× or 4× speed improvement while paying very little in terms of performance, as shown in Figure 5.6(a), as explained in Section 5.6.1 and Fig.5.5 .

5.8 Summary and limitations

Summary: We presented AMTNet and TraMNet two deep learning framework for action detection in videos. While AMTNet remains simple and efficient as it generates action cuboid proposals, whereas, TraMNet can cope with real-world videos containing "dynamic" actions whose location significantly changes over time. This is done by learning a transition probability matrix for each feature pyramid layer from the training data in a hidden Markov model formulation, leading to an original configurable layer architecture. Furthermore, unlike their competitors [15, 16], which require dense frame-level bounding box annotation, AMTNet and TraMNet builds on a similar network architecture in which action representations are learnt from pairs of frames rather than chunks of consecutive frames, thus eliminating the need for dense annotation. An extensive experimental analysis supports TraMNet's action detection capabilities, especially under dynamic actions and sparse annotations.

Limitations of TraMNet as oppose to AMTNet: We can argue that TraMNet is more generic and provides better spatiotemporal localisation performance. However there are few advantages to AMTNet, i) AMTNet is simple and does not require pre-computation of the transition matrix, ii) it has fewer hyperparameters, as it does not require a threshold to sample micro-tubes proposals from training data, iii) the time complexity of AMTNet is constant iv) in cases where there is not much movement, AMTNet might still be a more reliable option. We recommend trying AMTNet first on a new dataset before moving on to more complex TraMNet. As a result, we recommend using both methods in tandem.

Looking ahead: So far in this dissertation, we covered different methods of action tube detection in untrimmed videos in online fashion. Interestingly, similar to early action prediction problem observed in chapter 4 as a consequence of the online nature of action tube detection solution, we can look to predict the future of action tubes themselves. Any future prediction approach needs to online in nature, hence, we can use our existing online action tube detection methods for predicting the future of action tubes. In the next chapter, we will see how online tube construction along with future regression help in the future prediction of action tubes.

Chapter 6

Predicting Future of Action Tubes

6.1 Introduction

Imagine a pedestrian on the sidewalk, and an autonomous car cruising on the nearby road. If the pedestrian stays on the sidewalk and continues walking, they are of no concern for an autonomous vehicle. What, instead, if they start approaching the road, in a possible attempt to cross it? Any future prediction about the pedestrian's action and their possible position on/off the road would crucially help the autonomous car avoid any potential incident. It would suffice to foresee the pedestrian's action label and position half a second early to avoid a major accident. As a result, awareness about surrounding human actions and their future location are essential for the robot-car.

We can formalise the above problem as follows. We seek to predict both the class label and the future spatial location(s) of an action instance as early as possible, as shown in Figure 6.1. Basically, it translates into early spatiotemporal action detection (similar to [9] and Chapter 4) and future location prediction of tubes, achieved by completing action instance(s) for the unobserved part of the video. In earlier Chapter 4, we learned to predict the action class label early for an input video by observing a smaller portion (a few frames) of it, whilst the system incrementally builds action tubes in an online fashion. In contrast, here, we propose to predict both the class label of an action and its future location(s) (i.e., the future shape of an action tube). In this chapter, by *'prediction'* we refer to the estimation of both an action's label and location in *future*, unobserved video segments. An illustration of above is shown in Figure 6.1.

As observed in Chapter 5, the ability to predict *action micro-tubes* (sets of temporally connected bounding boxes spanning k video frames) from pairs of

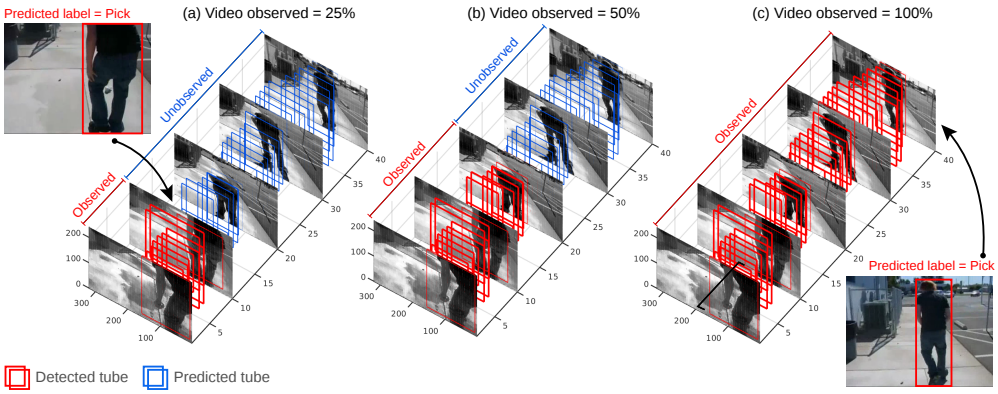


Figure 6.1: An Illustration of the action tube prediction problem using an example in which a "pickup" action is being performed on a sidewalk. As an ideal case, we want the system to predict an action tube as shown in (c) (i.e. when 100% of the video has been processed) just by observing 25% of the entire clip (a). We want the tube predictor to predict the action class label (shown in red) alongside predicting the spatial location of the tube. The red shaded bounding boxes denote the detected tube in the observed portion of the input video, whereas, the blue coloured bounding boxes represent the future predicted action tube for the unobserved part of the clip.

frames [24] or sets of k frames [15, 16] provides a powerful tool to extend the single frame-based online localisation approach discussed in Chapter 4 (i.e. [9]) to a action location prediction approach, while retaining its incremental nature. Combining the basic philosophies of [9] (Chapter 4) and [24] (Chapter 5) thus has the potential to provide an interesting and scalable approach to action prediction. As a result, we will build our tube prediction approach on two previous chapters. We can extend AMTNet by adding future prediction output heads to AMTNet to predict the future and past of each micro-tube. We call this new deep network a Tube Predictor Network ('TPNet'). We will show that such a network not only able to perform future tube prediction but also improve detection performance.

Related publications. The work presented in this chapter has appeared in ECCV workshop on anticipating human behaviour, 2018 [26]. The dissertation author is the primary investigator in [26]. Note, that this publication [26] and chapter built upon our previous works described in [9, 24] and two previous chapters.

Outline: The remaining chapter is organised as follows. We start by presenting an overview of our TPNet based tube prediction approach in Section 6.2. Next, we describe the change in AMNet as compared to the previous chapter in Section 6.3. In Section 6.4, we describe the training procedure of our TPNet along with multi-task learning objective in Section 6.4.1. Next in Section 6.5,

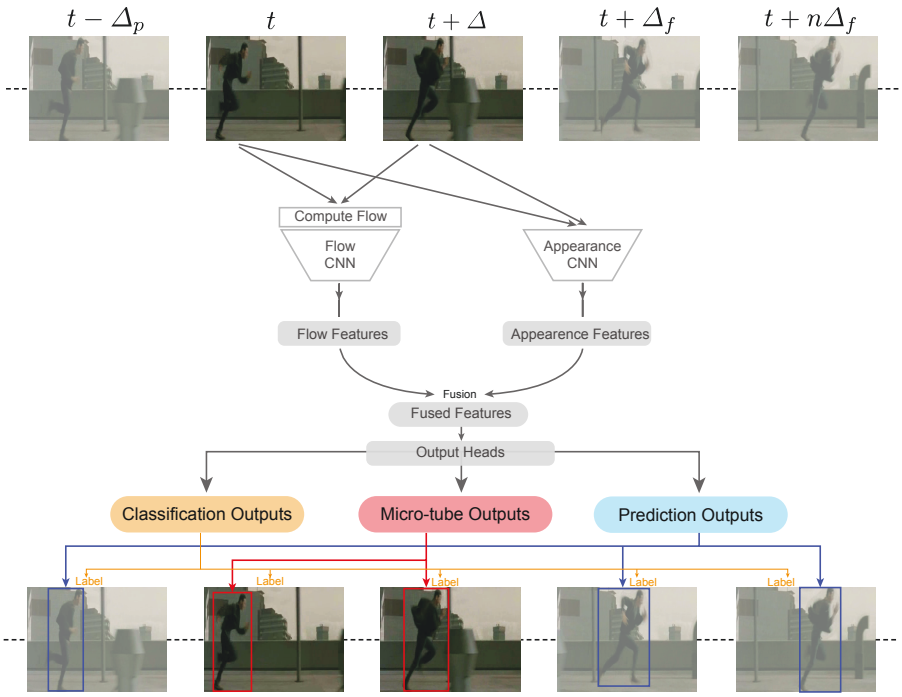


Figure 6.2: Workflow illustrating the application of TPNet to a test video at a time instant t. The network takes frames f_t and $f_{t+\Delta}$ as input and generates classification scores, the micro-tube (in red) for frames f_t and $f_{t+\Delta}$, and prediction bounding boxes (in blue) for frames $f_{t-\Delta_p}$, $f_{t+\Delta_f}$ up to $f_{t+n\Delta_f}$. All bounding boxes are considered to be linked to the micro-tube. Note that predictions also span the past: a setting called smoothing *in the estimation literature. n, Δ_f, and Δ_p are network parameters that we cross-validate during training.*

we present how to use TPNet in a sliding window fashion in the temporal direction while generating micro-tubes and corresponding future predictions. These, eventually, are fed to a tube prediction framework (§ 6.5) to generate the future of any current action tube being built using micro-tubes. Then, we present the evaluation of our TPNet based approach in Section 6.6. Finally, we present a summary in Section 6.7.

6.2 Overview of the approach

We propose to extend the action micro-tube detection architecture presented in previous chapter to produce, at any time t, past ($\tau < t$), present, and future ($\tau > t$) detection bounding boxes, so that each (extended) micro-tube contains bounding boxes for both observed and not yet observed frames. All bounding boxes, spanning presently observed frames as well as past and future ones (in which case we call them predicted bounding boxes), are considered to be linked, as shown in blue in Figure 6.2.

Further, in virtue of TPNet and online tube construction, the temporally

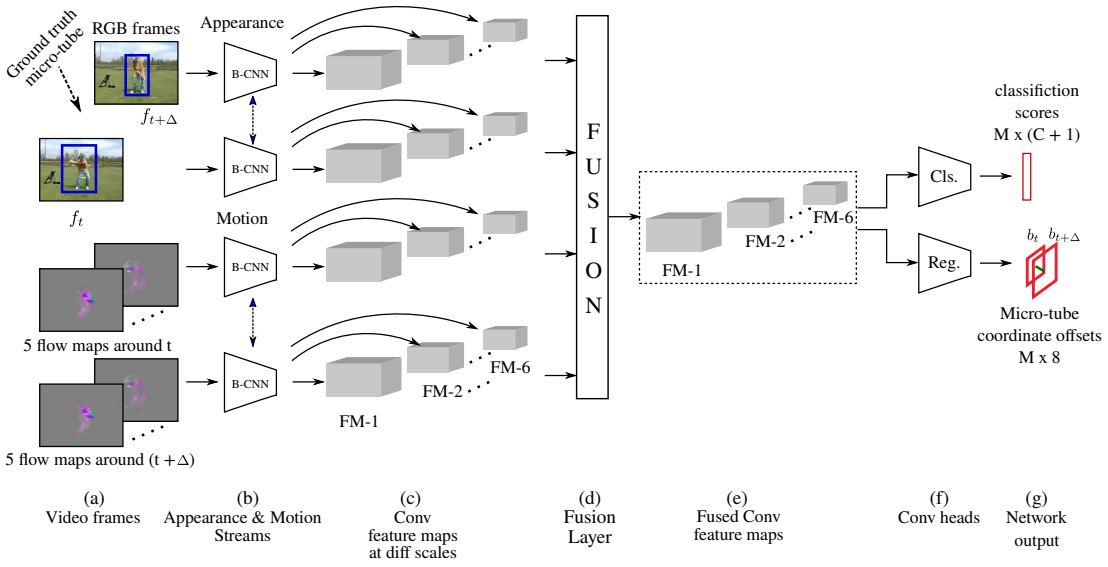


Figure 6.3: Overview of the action micro-tube detection network (AMTnet). Here we introduce feature-level fusion (d).

linked micro-tubes forming each currently detected action tube (spanning the observed segment of the video) also, contain past and future estimated bounding boxes. As these predicted boxes are implicitly linked to the micro-tubes which compose the presently detected tube, the problem of linking to the latter the future bounding boxes, leading to a whole action tube, is automatically addressed.

6.2.1 Resulting contributions

In summary, we present a Tube Predictor network (TPNet) which:

- given a partially observed video, can (early) predict video long action tubes in terms of both their classes and the constituting bounding boxes;
- demonstrates that training a network to make predictions also helps in improving action detection performance;
- demonstrates that feature-based fusion works better than late fusion in the context of spatiotemporal action detection.

6.3 Base network

Our TPNet build on AMTNet in previous chapter. However, we make some improvement on the fusion side, earlier we performed a late fusion of appearance and motion cues. Here, in this chapter, we introduce a feature-level fusion

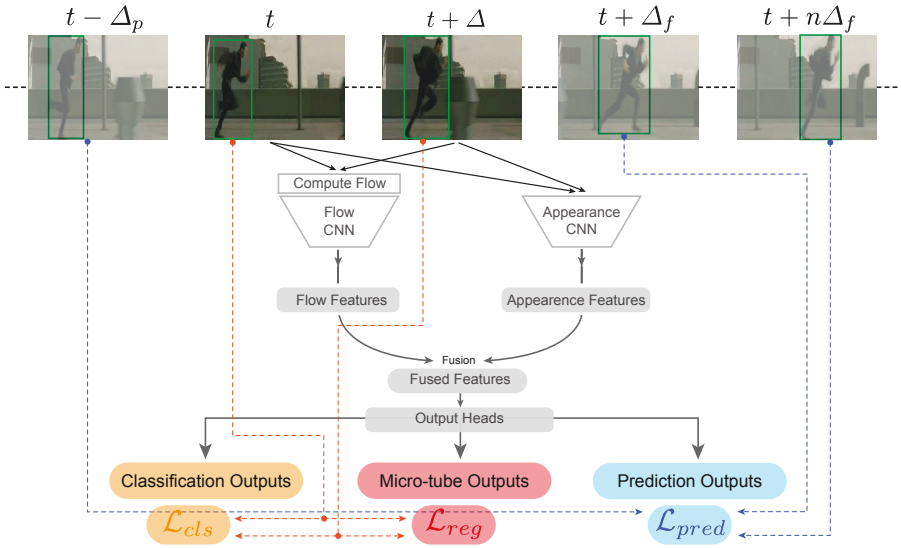


Figure 6.4: Overview of the tube predictor network architecture at training time.

of appearance and motion cues for AMTNet network. An overview of AMTNet with fusion is shown in Figure 6.3 We try two different types (summation and concatenation) of fusion methods. We show that both methods perform similarly, however, frame-level fusion provides a big performance improvement over late-fusion techniques used in previous chapters on JHMDB-21 dataset.

6.3.1 Converting AMTNet into TPNet

AMTNet allows us to generate micro-tubes which has to connected boxes from a pair of frames at time t and $t+\Delta$. While observing the same pair frames, we can extend the idea of connected boxes further by predicting a box for frame time $t+\Delta_f$. We can take it even further, we can predict connected boxes for n future frames which we haven't observed, also even for a past frame $t-\Delta_p$. Note that predictions also span the past: a setting called *smoothing* in the estimation literature. We only observe a pair of frames from time t and $t+\Delta$, and Δ_p measure how far in the past we are predicting into, whereas Δ_f is a future step size, and n is the number of future steps. Figure 6.4 show an overview of tube predictor network (TPNet), which incorporate above prediction ideas. Based on this idea we will now explain how to train such a network in the next Section 6.4, and for prediction in Section 6.5.

6.4 Training TPNet

The underlying architecture of TPNet is shown in Figure 6.4. TPNet takes two successive frames from time t and $t+\Delta$ as input. The two input frames are fed

to two parallel CNN streams, one for appearance and one for optical flow. The resulting feature maps are fused, either by summation or concatenation of the given feature maps. Finally, three types of convolutional output heads are used for P prior boxes as shown in Figure 6.4. The first one produces the $P \times (C+1)$ classification outputs; the the second one regresses the $P \times 8$ coordinates of the micro-tubes, as in AMTnet; the last one regresses $P \times (4(1+n))$ coordinates, where 4 coordinates correspond to the frame at $t - \Delta_p$, and the remaining $4n$ are associated with the n future steps. The training procedure for the new architecture is illustrated below.

6.4.1 Multi-task learning

TPNet is designed to strive for three objectives, for each prior box p. The first task (i) is to classify the P prior boxes; the second task (ii) is to regress the coordinates of the micro-tubes; the last (iii) is to regress the coordinates of the past and future detections associated with each micro-tube.

Given a set of P anchor boxes and the respective outputs we compute a loss following the training objective of SSD [21]. Let $x^c_{i,j} = \{0, 1\}$ be the indicator for matching the i-th prior box to the j-th ground truth box of category c. We use the bipartite matching procedure described in [21] for matching the ground truth micro-tubes $G = \{g_t, g_{t+\Delta}\}$ to the prior boxes, where g_t is a ground truth box at time t. The overlap is computed between a prior box p and micro-tube G as the mean IoU between p and the ground truth boxes in G. A match is defined as positive ($x^c_{i,j} = 1$) if the overlap is more than or equal to 0.5.

The overall loss function $\mathcal{L}$ is the following weighted sum of classification loss ($\mathcal{L}_{cls}$), micro-tube regression loss ($\mathcal{L}_{reg}$) and prediction loss ($\mathcal{L}_{pred}$):

$$\mathcal{L}(x, c, m, G, z, Y) = \frac{1}{N}\big(\mathcal{L}_{cls}(x, c) + \alpha\mathcal{L}_{reg}(x, m, G) + \beta\mathcal{L}_{pred}(x, z, Y)\big), \quad (6.1)$$

where N is the number of matches, c is the ground truth class, m is the predicted micro-tube, G is the ground truth micro-tube, z assembles the predictions for the future and the past, and Y is the ground truth of future and past bounding boxes associated with the ground truth micro-tube G. The values of α and β are both set to 1 in all of our experiments: different values might result in better performance.

The classification loss $\mathcal{L}_{cls}$ is a softmax cross-entropy loss; a hard negative mining strategy is also employed, as proposed in [21]. The micro-tube loss $\mathcal{L}_{reg}$ is a smooth L1 loss [81] between the predicted (m) and the ground truth (G) micro-tube. Similarly, the prediction loss $\mathcal{L}_{pred}$ is also a smooth L1 loss

between the predicted boxes (z) and the ground truth boxes (Y). As in [21,81], we regress the offsets with respect to the coordinates of matched prior box p matched to G for both m and z. We use the same offset encoding scheme as used in [21].

6.5 Tube prediction framework

6.5.1 Problem statement

Similar to previous chapters, we define an *action tube* as a connected sequence of detection boxes in time without interruptions and associated with a same action class c, starting at first frame f_1 and ending last frame f_T, in trimmed video: $\mathcal{T}_c = \{b_1, ...b_t, ...b_T\}$. Tubes are constrained to span the entire video duration, like in [8]. At any time point t, a tube is divided into two parts, one needs to be detected $\mathcal{T}_c^d = \{b_1, ...b_t\}$ up to f_t and another part needs to be predicted/estimated $\mathcal{T}_c^p = \{b_{t+1}, ...b_T\}$ from frame f_{t+1} to f_T along with its class c. The observed part of the video is responsible for generating $\mathcal{T}_c^d$ (red in Fig 6.1), while we need to estimate the future section of the tube $\mathcal{T}_c^p$ (blue in Fig 6.1) for the unobserved segment of the video. The first sub-problem, the online detection of $\mathcal{T}_c^d$, is explained in Section 5.6 of Chapter 5. The second sub-problem (the estimation of the future tube segment $\mathcal{T}_c^p$) is tackled by a tube predictor network (TPNet, § 6.4) in a novel tube prediction framework describe in next Section 6.5.2.

6.5.2 Tube prediction using TPNet

TPNet is shown in Figure 6.2 at test time. Similar to the training setting, TPNet observes only two frames that are Δ apart at any time point t. The outputs of TPNet at any time t are linked to a micro-tube, each micro-tube containing a set of bounding boxes $\{m_t = \{b_t, b_{t+\Delta}\}; z_t = \{b_{t-\Delta_p}, b_{t+\Delta_f}, ..., b_{t+\Delta_f}\}\}$, which are considered as linked together.

As explained in Section 5.6 of Chapter 5, given a set of micro-tubes $\{m_1...m_{t-\Delta}\}$, we can construct $\mathcal{T}_c^d$ by online linking [9] of the micro-tubes. As a result, we can use predictions for $t + \Delta_f$ up to $t + n\Delta_f$ to generate the future of $\mathcal{T}_c^d$, thus extending it further into the future as shown in Figure 6.5. More specifically, as it is indicated in Figure 6.5(a), a micro tube at $t - 2\Delta$ is composed by $n + 2$ bounding boxes ($\{b_{t-2\Delta}, b_{t-\Delta}, b_{t-2\Delta+\Delta_f}, ...b_{t-\Delta+n\Delta_f}\}$) linked together. The last micro-tube is generated from $t - \Delta$. In the same fashion, putting together the predictions associated with all the past micro-tubes ($\{m_1...m_{t-\Delta}\}$) yields

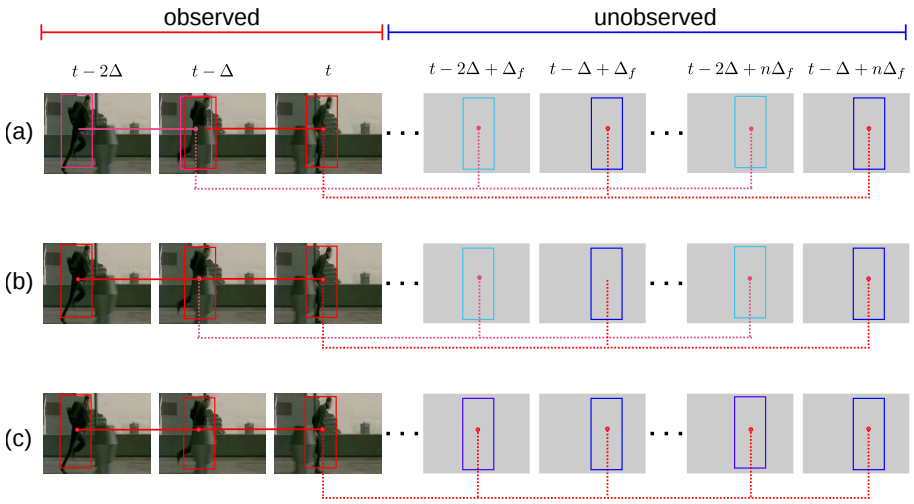


Figure 6.5: Overview of future tube ($\mathcal{T}_c^p$) prediction using the predictions that are linked to micro-tubes. The first row (a) shows two output micro-tubes in light red and red and their corresponding predictions in future in light blue and blue. In row (b) two micro-tubes are linked together, after which they are shown in the same colour (red). By induction on the previous step, in row (c) we show that the predictions associated with two micro-tubes are linked together as well, hence forming one single tube. The observed segment is shown in red, while the predicted segment for the part of the video yet to observe is shown in blue in case of prediction. There may be an overlap of predictions for the same tube, precedence to use the box in final tube is defined by: detection box >> most recent prediction >> older prediction box. It is explained in more details at the end of Section 6.5.2.

a set of linked future bounding boxes ($\{b_{t+1}, ..., b_{t+\Delta+\Delta_f}, ..., b_{t-\Delta+n\Delta_f}\}$) for the current action tube $\mathcal{T}_c^d$, thus outputting a part of the desired future $\mathcal{T}_c^p$.

Now, we can generate future tube $\mathcal{T}_c^p$ from the set of linked future bounding boxes ($\{b_{t+1}, ...b_{t-\Delta+\Delta_f}, ...b_{t-\Delta+n\Delta_f}\}$) from $t+1$ to $t-\Delta+n\Delta_f$ and simple linear extrapolation of bounding boxes from $t-\Delta+n\Delta_f$ to T. Linear extrapolation is performed based on the average velocity of the each coordinates from last 5 frames, predictions outside the image coordinate are trimmed to the image edges.

There may be an overlap of predictions for the same tube, for instance, if both Δ_f and Δ equal to 1 then at time t there a future predicted box from $t-1$ and a detected box from t, in that case, detection takes precedence over future prediction, i.e detected box is used in final tube. In any of such case, precedence to use the box in final tube is defined by: detection box >> most recent prediction >> older prediction box.

6.6 Experiments

We test our action tube prediction framework (§ 6.5) on four challenging problems:
i) action detection (§ 6.6.1),
ii) early action prediction (§ 6.6.2),
iii) online action detection (§ 6.6.2),
iv) future action tube prediction (§ 6.6.3)
Finally, evidence of real time capability is quantitatively demonstrated in (§ 6.6.4).

Evaluation protocal. Now, we define the evaluation metrics used in the current chapter for the tasks mentioned above. i) We use a standard mean-average precision metric to evaluate the detection performance when the whole video is observed. ii) Early label prediction task is evaluated by video classification accuracy [9, 108] as early as when only 10% of the video frames are observed.
iii) Online action detection (§ 6.6.2) is set up based on the experimental setup of [9], and use mAP (mean average precision) as metric for online action detection i.e. it evaluates present tube ($\mathcal{T}_c^d$) built-in online fashion.
iv) The future tube prediction is a new task; we propose to evaluate its performance in two ways, one completion-mAP (c-mAP) and prediction-mAP (p-mAP). These meterics are define in Section 2.10.

Similar to Chapter 4, we report the performance of previous three tasks (i.e. task ii to iv) as a function of *Video Observation Percentage*, i.e., the portion (%) of the entire video observed.

J-HMDB-21. We evaluate our model on the J-HMDB-21 [10] dataset. Each video contains an instance of an atomic action for 20-40 frames. Although, videos are of short duration (max 40 frames), we consider this dataset because tubes belong to the same class and we think it is a good dataset to start with for action prediction task. More details about the dataset can be found in Section 2.9.

Baseline. We modified AMTnet to fuse flow and appearance features 6.3. We treat it as a baseline for all of our tasks. Firstly, we show how feature fusion helps AMTnet in Section 6.6.1, and compare it with other action detection methods along with our TPNet. Secondly in Section 6.6.3, we linearly extrapolate the detection from AMTnet to construct the future tubes and use them as a baseline for the tube prediction task.

TPNet$_{abc}$. The training parameters of our TPNet are used to define the name of the setting in which we use our tube prediction network. The network name TPNet$_{abc}$ represents our TPNet where $a = \Delta_p$, $b = \Delta_f$ and $c = n$, if Δ_p is set

Table 6.1: Action detection results on JHMDB dataset. The table is divided into four parts. The first part lists approaches which takes a single frame as input; the second part presents approaches which takes multiple frames as input; the third part contemplates different fusion strategies of our feature-level fusion (based on AMTnet); lastly, we report the detection performance of our TPNet by ignoring the future and past predictions and only use the detected micro-tubes to produce the final action tubes.

Methods	$\delta = 0.2$	$\delta = 0.5$	$\delta = 0.75$	$\delta = .5{:}.95$	Acc %
MR-TS Peng *et al.* [90]	74.1	73.1	–	–	–
FasterRCNN Saha *et al.* [14]	72.2	71.5	43.5	40.0	–
OJLA Behl *et al.* [109]*	–	67.3	–	36.1	–
SSD Singh *et al.* [9]*	73.8	72.0	44.5	41.6	–
AMTnet Saha *et al.* [24] rgb-only	57.7	55.3	–	–	–
ACT kalogeiton *et al.* [15]*	74.2	73.7	52.1	44.8	61.7
T-CNN (offline) Hou *et al.* [16]	**78.4**	76.9	–	–	67.2
MR-TS [90] + I3D [17] Gu *et al.* [11]	–	**78.6**	–	–	–
AMTnet-LateFusion*	71.7	71.2	49.7	42.5	65.8
AMTnet-FeatFusion-Concat*	73.1	72.6	59.8	48.3	68.4
AMTnet-FeatFusion-Sum*	73.5	72.8	59.7	48.1	69.6
Ours TPNet_{053}*	72.6	72.1	58.0	46.7	67.5
Ours TPNet_{453}*	73.8	73.0	59.1	47.3	68.2
Ours TPNet_{051}*	74.6	73.1	60.5	49.0	**69.8**
Ours TPNet_{451}*	74.8	74.1	**61.3**	**49.1**	68.9

TPNet_{abc} represents our TPNet where $a = \Delta_p$, $b = \Delta_f$ and $c = n.$; * means online methods

to 0 it means network doesn't learn to predict the past bounding boxes. In all of our settings, we use $\Delta = 1$.

Implementation details. We train all of our networks with the same set of hyper-parameters to ensure fair comparison and consistency, including TPNet and AMTnet. We use an initial learning rate of 0.0005, and the learning rate drops by a factor of 10 after $5K$ and $7K$ iterations. All the networks are trained up to $10K$ iterations. We implemented AMTnet using PyTorch (`https://pytorch.org/`). We initialise AMTnet and TPNet models using the pretrained SSD network on J-HMDB-21 dataset on its respective train splits. The SSD network training is initialised using image-net trained VGG network. For, optical flow images, we used optical flow algorithm of Brox *et al.* [22]. Optical flow output is put into a three-channel image, two channels are made of flow vector and the third channel is the magnitude of the flow vector.

6.6.1 Action detection performance

Table 6.1 shows the traditional action detection results for the whole action tube detection in the videos of J-HMBD-21 dataset.

Feature fusion compared to the late fusion scheme in AMTnet shows (Table 6.1) remarkable improvement, at detection threshold $\delta = 0.75$ the gain with

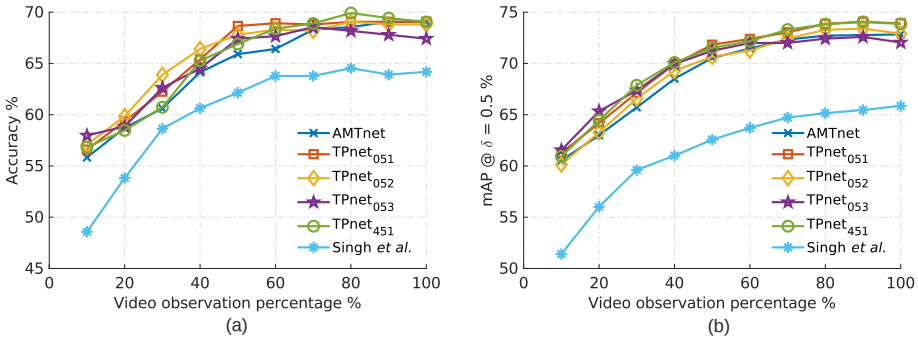


Figure 6.6: Early label prediction results (video-level label prediction accuracy) on J-HMDB-21 dataset in sub-figure (a). Online action detection results (mAP with detection threshold $\delta = 0.5$) on J-HMDB-21 dataset are shown in sub-figure (b). $TPNet_{abc}$ represents our TPNet where $a = \Delta_p$, $b = \Delta_f$ and $c = n$.

feature level fusion is 10%, as a result, it is able to surpass the performance of ACT [15], which relies on a set of 6 frames as compared to AMTnet which uses only 2 successive frames as input. Looking at the average-mAP ($\delta = 0.5 : 95$), we can see that the fused model improves by almost 8% as compared to single frame SSD model of Singh *et al.* [9]. We can see that concatenation and sum fusion perform almost similar for AMTnet. Sum fusion is little less memory intensive on the GPUs as compared to the concatenation fusion; as a result, we use sum fusion in our TPNet.

TPNet for detection is shown in the last part of the Table 6.1, where we only use the detected micro-tubes by TPNet to construct the action tubes. We train TPNet to predict future and past (i.e. when $\Delta_p > 0$) as well as present micro-tubes. We think that predicting bounding boxes for both the past and future video segments act as a regulariser and helps improving the representation of the whole network. Thus, improving the detection performance (Table 6.1 TPNet_{051} and TPNet_{451}). However, that does not mean adding extra prediction task always help when a network is asked to learn prediction in far future, as is the case in TPNet_{053} and TPNet_{453}, we have a drop in the detection performance. We think there might be two possible reasons for this, i) network might starts to focus more on the prediction task, and ii) videos in J-HMDB-21 are short and the number of training samples decreases drastically ($19K$ for TPNet_{051} and $10K$ for TPNet_{453}), because we can not use edge frames of the video in training samples as we need a ground truth bounding box which is 15 frames in the future, as $\Delta_f = 5$ and $n = 3$ for TPNet_{053}. However, in Section 6.6.3, we show that the TPNet_{053} model is the best to predict the future very early.

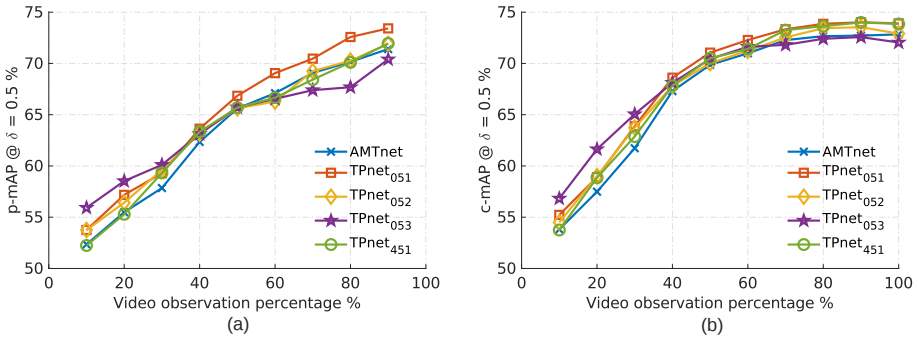


Figure 6.7: Future action tube prediction results (a) (prediction-mAP (p-mAP)) for predicting the tube in unobserved part of the video. Action tube prediction results (b) (completion-mAP (c-mAP)) for predicting video long tubes as early as possible on J-HMDB-21 dataset in sub-figure (b). We use p-mAP (a) and c-mAP (b) with detection threshold $\delta = 0.5$ as evaluation metrics on J-HMDB-21 dataset. TPNet$_{abc}$ represents our TPNet where $a = \Delta_p$, $b = \Delta_f$ and $c = n$.

6.6.2 Early label prediction and online detection

Figure 6.6 (a) & (b) show the early prediction and online detection capabilities of Singh *et al.* [9], AMTnet-Feature Fusion-sum and our TPNet.

Soomro *et al.* [108]'s method also perform early label prediction on J-HMDB-21; however, their performance is deficient, as a result, the plot would become skewed (Figure 6.6(a)), so we omit theirs from the figure. For instance, by observing only the initial 10% of the videos in J-HMDB-21, TPNet$_{453}$ able to achieve a prediction accuracy of 58% as compared to 48% by Singh *et al.* [9] and 5% by Soomro *et al.* [108], which is in fact higher than the 43% accuracy achieved by [108] after observing the *entire* video. As more and more video observed, all the methods improve, but TPNet$_{451}$ show the most gain, however, TPNet$_{053}$ observed the least gain from all the TPNet settings shown. Which is in-line with action detection performance discussed in the previous section 6.6.1. We can observe the similar trends in online action detection performance shown in Figure 6.6(b). To reiterate, TPNet$_{053}$ doesn't get to see the training samples from the end portion of the videos, as it needs a ground truth bounding box from 15 frames ahead. So, the last frame it sees of any training video is $T-15$, which is almost half the length of the most extended video(40 frames) in J-HMDB-21.

6.6.3 Future action tube prediction

Our main task of this work is to predict the future of action tubes. We evaluate it using two newly proposed metrics (*p-mAP* and *c-mAP*) as explained earlier at the start of the experiment section 6.6. Result are shown in Figure 6.7 for

future tube prediction (Figure 6.7 (a)) with p-mAP metric and tube completion with c-mAP as metric.

Although, the TPNet_{053} is the worst setting of TPNet model for early label prediction (Fig. 6.6(a)), online detection(Fig. 6.6(b)) and action tube detection (Table 6.1), but as it predicts furthest in the future (i.e. 15 frame away from the present time), it is the best model for early future tube prediction (Fig. 6.7(a)). However, it does not observe as much appreciation in performance as other settings as more and more frames are seen, owing to the reduction in the number of training samples. On the other hand, TPNet_{451} observed large improvement as compared to TPNet_{051} as more and more portion of the video is observed for tube completion task (Fig.6.7(b)), which strengthen our argument that predicting not only the future but also, the past is useful to achieve more regularised predictions.

Comparison with the baseline. As explained above, we use AMTnet as a baseline, and its results can be seen in all the plots and the Table. We can observe that our TPNet performs better than AMTnet in almost all the cases, especially in our desired task of early future prediction (Fig 6.7(a)) TPNet_{043} shows almost 4% improvement in p-mAP (at 10% video observation) over AMTnet.

Discussion. Predicting further into the future is essential to produce any meaningful predictions (seen in TPNet_{053}), but at the same time, predicting past is helpful to improve overall tube completion performance. One of the reasons for such behaviour could be that J-HMDB-21 tubes are short (max 40 frames long). We think training samples for a combination of TPNet_{053} and TPNet_{451}, i.e. TPNet_{453} are chosen uniformly over the whole video while taking care of the absence of ground truth in the loss function could give us better of both settings. The idea of regularising based on past prediction is similar to the one used by Ma *et al.* [50], where they show that regularisation based on past can help in the prediction of the future.

6.6.4 Test Time Detection Speed

Continue from previous chapters, here, we show real-time capabilities. Here we use their online tube generation method from Chapter 4 for our tube prediction framework to inherit online capabilities. The only question mark is TPNet's forward pass speed. We thus measured the average time taken for a forward pass for a batch size of 1 as compared to 8 by [9]. A single forward pass takes 46.8 milliseconds to process one example, showing that it can be run in almost real-time at 21fps with two streams on a single 1080Ti GPU. One can improve

speed even further by testing TPNet with Δ equal to 2 or 4 and obtain a speed improvement of 2× or 2×. However, use of dense optical flow [22], which is slow, but as in [9], we can always switch to real-time optical [23] with small drop in performance.

6.7 Summary and limitations

Summary: We presented TPnet, a deep learning framework for future action tube prediction in videos which, unlike previous online tube detection methods [9, 108], generates future of action tubes as early as when 10% of the video is observed. It can cope with future uncertainty better than the baseline methods while remaining state-of-the-art in action detection task. Hence, we provide a scalable platform to push the boundaries of action tube prediction research; it is implicitly scalable to multiple action tube instances in the video as future prediction is made for each action tube separately. We plan to scale TPnet for action prediction in temporally untrimmed videos in the future.

Limitations: The current tube prediction method is limited in the following ways: i) it work on small videos which contain only one action instance each, ii) training is sub-optimal when prediction needs to be made for the long term because videos are small. iii) feature representation only depends on two frames separated by Δ steps.

Looking ahead: We presented action detection and prediction framework in previous chapters which solely rely on frame-level 2D CNNs for feature representation, which is required for speed purpose. However, 3D representation from a continuous set of frames is state-of-the-art in many video understanding tasks, as discussed in the related work chapter (Chapter 2). In the next chapter, we will present a 3D representation method which is causal/online in nature. These online methods are important for any online action detection/prediction approach to be useful in real-world applications.

Part III : Online/Causal 3D Representations

Chapter 7

Recurrent Convolution for Causal 3D CNNs

7.1 Introduction

Convolutional neural networks (CNN) are starting to exhibit gains in action recognition from videos similar to those previously observed in image recognition [45, 96] thanks to new 3D CNNs [17–19, 57, 149]. For instance, Hare *et al.* [149] have shown that that is the case for the 3D version of 2D residual networks (ResNets) [148]. However, they have multiple problem concerning their deployment in online video understanding problems, e.g., online action detection [9, 108], future action label/tube prediction [26, 111], and future representation prediction [116].

Firstly, temporal convolutions are inherently anti-causal, which is necessary for online video understanding. Secondly, the temporal convolution size needs to be picked by hand at every level of network depth. As a result, the temporal reasoning horizon or 'receptive field' is effectively constrained by the size of the temporal convolution kernel(s). Lastly, 3D CNNs do not preserve temporal resolution, as the latter drops with network depth. Preserving temporal resolution, in opposition, is essential in problems such where we need predictions to be made on each frame of input clip while reasoning about temporal context, e.g. bounding box regression on each frame for action tube detection [8, 9] or temporal label prediction on each frame for temporal action segmentation [4, 29] or online video segmentation [150].

Our method: combining implicit and explicit temporal modelling. Inspired by the success of 3D CNNs, we propose to make 3DCNNs causal by explicit temporal modelling. Hidden state models, such as Markov ones [151], recurrent neural networks (RNN) [65, 152], and long short-term memory (LSTM)

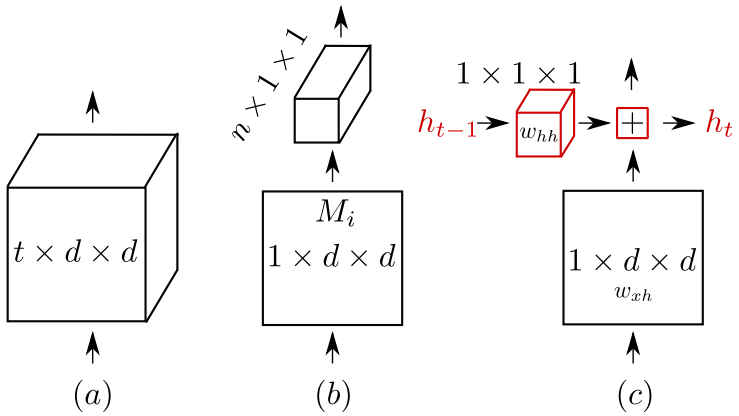


Figure 7.1: Illustration of basic 3D units used to build 3D networks to process sequences of input frames. (a) Standard 3D convolution, as in I3D [17] or C3D [53]. (b) 3D convolution decomposed into a 2D spatial convolution followed by a 1D temporal one, as in S3D [18]. In R(2+1)D [19] the number M_i of middle planes is increased to match the number of parameters in standard 3D convolution. (c) Our proposed decomposition of 3D convolution into 2D spatial convolution and recurrence (in red) in the temporal direction, with a $1 \times 1 \times 1$ convolution as hidden state transformation.

networks [66] can all be used to model temporal dynamics in videos [48, 153], allowing flexible temporal reasoning.

In an approach which aims to combine the representation power of explicit dynamical models with the discriminative power of 3D networks, in this work we propose a recurrent alternative to 3D convolution illustrated in Figure 7.1(c). In this new architecture, spatial reasoning, expressed in the form of a mapping from the input to a hidden state, is performed by spatial convolution (with kernel size $1 \times d \times d$), whereas temporal reasoning (represented by hidden state-to-hidden state transformations) is performed by a convolution (with kernel size $1 \times 1 \times 1$) taking place at every point in time and at each level (depth-wise) of the network. In a setting which combines the effect of both operators, the hidden state at time t (denoted by h_t) is a function of both the output of the spatial convolution and of the output of the temporal (hidden) convolution with h_{t-1} as an input.

As a result, the temporal reasoning horizon is effectively unconstrained, as the hidden state h_t is a function of the input in the interval $[0, t]$. Also, we are able to convert anti causal 3DCNNs into causal one while preserving the temporal resolution.

To complete the transformation of 3DCNNs into causal 3DCNNs, we need to take care of some limitations of RNNs, such as vanishing/exploding gradients problem and activation functions (which could slow down the training). Interestingly, Le *et al.* [154] show that simple RNNs can exhibit long-term memory properties if appropriately initialised, even better than LSTMs. They use

ReLU [155] activation functions because of their fast convergence and sparsity properties [155], as opposed *sigmoid* or *tanh* activation function used with traditional RNNs. The use of ReLU is in line with standard practice in CNNs. We thus follow [154] and initialise our hidden-to-hidden convolution (kernel size $N \times N \times 1 \times 1 \times 1$) by the identity matrix, where N is number of hidden state kernels. Spatial convolution, instead, can simply be initialised using ImageNet pre-trained weights. We will show that such initialisation is important to achieve competitive performance to that of 3DCNNs.

Related publications. The work presented in this chapter [27] is accpeted for publication at ICCV workshop on sarge scale video understanding. The dissertation author is the primary investigator in [27].

Outline: The rest of the chapter is organised as follow: we start with the overview of our approach in Section 7.2. We explain how 3D CNNs are created fron 2D CNNs in Section 7.3 for baseline purpose. Next, we introduce our proposed casual 3D CNN architecture in Section 7.4, including its properties. In Section 7.5, we validate our approach with various studies on two large scale dataset. Finally, we present a short summary and in Section 7.6.

7.2 Overview of the approach

Our approach builds upon existing 3D CNNs, which are described in Section 7.3. We replace every 3D convolution (Fig. 7.1(a)) in 3D CNNs (§ 7.3) with our Recurrent Convolutional Units (RCU) (Fig. 7.1(c) - § 7.4.1). As, a result, a 3D CNN is converted into a casual 3D CNNs, named RCN (§ 7.4.2). Next, it is important to start from good initialisation, to this, we proposed to initialise our RCN with two separate initialisation process. First, each 2D convolution is initialised with pre-trained 2D network weights (§ 7.4.5). Next, hidden state convolution matrix weight are initialised with identity matrix, as explained in Section 7.4.6.

7.2.1 Resulting contributions

We present a new approach to video feature representation based on an original convolutional network with recurrent hidden states at each depth level, which:

- allows flexible temporal reasoning, as it exploits by design information coming from all the input sequence observed up to time t;

- generates output representations in a causal way, allowing online video processing and enabling the use of 3D networks in scenarios in which causality is key;

- is designed to directly benefit from model initialisation via ImageNet pretrained weights, as opposed to state of the art approaches, and in line with clear emerging trends in the field.

In our experiments, we show that our proposed RCN outperforms baseline I3D and (2+1)D models, while displaying all the above desirable properties.

7.3 2D to 3D CNNs

There are two main reason why 3D CNNs [17–19,57] evolved from 2D CNNs [49, 96] perform better than 3D CNNs built from scratch [52,53]. Firstly, 2D CNNs are well tried and tested on the problem of image recognition and a video is, after all, a sequence of images – hence, transferability makes sense. Secondly, initialisation from a good starting guess leads to better convergence in videos [17], since the number of parameters in 3D networks is huge.

In this section, we recall the two basic types of 3D CNNs that can be built using a 2D CNN architecture. We will use them as baselines in our experiments.

7.3.1 Inflated 3D network (I3D)

A 2D network can be converted/inflated into a 3D one by replacing a 2D $(d \times d)$ convolution with a 3D $(n \times d \times d)$ convolution as shown in Figure 7.1(a). Usually, the kernel's temporal dimension n is set to be equal to the spatial dimension d, as in the inflated 3D network (I3D) [17] or the convolutional 3D network (C3D) [53].
Here, we inflate the 18 layer ResNet [148] network into an I3D one as shown in Table 7.1, where each 2D convolution is inflated into a 3D convolution. Similarly to the I3D network in [17], a convolutional layer is used for classification, instead of the fully connected layer used in [19, 57]. A convolutional classification layer allows us to evaluate the model on sequences of variable length at test time. In this way, video-level results can be obtained in a seamless fashion as compared to computing clip-level outputs in sliding window fashion to obtain the video-level output.

Table 7.1: I3D ResNet-18 network architecture with its outputs sizes and RCN's output sizes for input with $16 \times 112 \times 112$ *size. Each Convolution layer of the network is defined by temporal (n) and spatial (d) size of kernel and number of kernels. ConvC layer uses the number of classes as the number of kernels.*

Layers		Output Sizes	
Names	n, d, number of kernels	I3D	RCN
conv1	$3, 7, 64$; stride $1, 2, 2$	$16 \times 56 \times 56$	$16 \times 56 \times 56$
res2	$[3, 3, 64\ \&\ 3, 3, 64] \times 2$	$16 \times 56 \times 56$	$16 \times 56 \times 56$
res3	$[3, 3, 128\ \&\ 3, 3, 128] \times 2$	$8 \times 28 \times 28$	$16 \times 28 \times 28$
res4	$[3, 3, 256\ \&\ 3, 3, 256] \times 2$	$4 \times 14 \times 14$	$16 \times 14 \times 14$
res5	$[3, 3, 512\ \&\ 3, 3, 512] \times 2$	$2 \times 7 \times 7$	$16 \times 7 \times 7$
pool	spatial pooling	$2 \times 1 \times 1$	$16 \times 1 \times 1$
convC	classification; $1, 1, C$	$2 \times C$	$16 \times C$
mean	temporal pooling	C	C

7.3.2 Separated convolution networks

Figure 7.1(b) shows how a 3D ($t \times d \times d$) convolution can be decomposed into a ($1 \times d \times d$) spatial convolution and a ($t \times 1 \times 1$) temporal one. Usually, the size of the temporal kernel t is set to be equal to its spatial dimension d, as in both I3D [17] and C3D [53].

Such a separated convolutional network (S3D) was introduced by Xie *et al.* [18]. The authors showed that such a decomposition not only reduces the number of parameters, but also delivers performances very much similar to those of traditional 3D CNNs. After taking a closer look at 3D convolution separation, Tran *et al.* [19] argued that if the number of kernels M_i used in spatial convolution (Figure 7.1(b)) are increased in such way that the numbers of parameters of spatial and temporal convolution combined are equal to the number of parameters in 3D convolution, then performance actually improves over 3D networks. However, such a change in the number of kernels does not allow initialisation from ImageNet pre-trained models any longer. They refer to their model as $(2+1)D$ model. Although the latter can be considered a special case of Pseudo-3D networks (P3D) [55] models, because of its homogeneity and simplicity the (2+1)D model performs better than P3D.

We re-implemented (2+1)D without ImageNet initialisation as an additional baseline as it has the most promising result without any additional trick like gating in S3Dg (S3D with gating).

7.4 3D Recurrent convolutional network

We are now ready to describe the architecture of our causal *Recurrent Convolutional (3D) Network* (RCN) and its properties in detail. Firstly, we show how

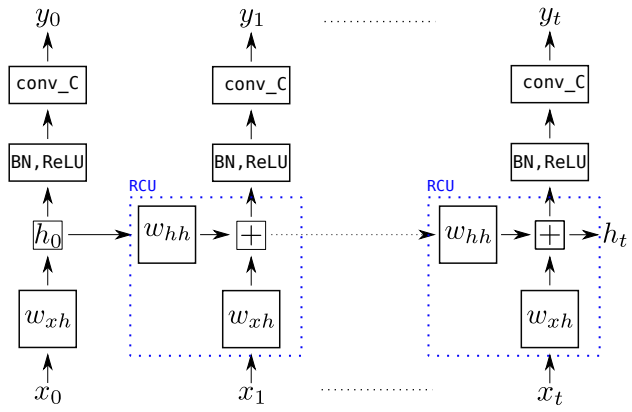


Figure 7.2: An unrolled version of Recurrent Convolutional Network (RCN) composed by a single RCU layer followed by a batch normalisation (BN) layer, a ReLU activation layer, and a final convolutional layer used for classification.

Recurrent Convolutional Units (RCUs) (§ 7.4.1) are used to replace *every* 3D convolutions in the I3D network (§ 7.3.1), resulting in our RCN model (§ 7.4.2). Next, we show how RCUs preserve temporal resolution in Section 7.4.3. Then in Section 7.4.4, we show how our network behaves in a causal manner. Lastly, in § 7.4.5 and § 7.4.6, we illustrate the initialisation process for RCN and RCU.

7.4.1 Recurrent convolutional unit

A pictorial illustration of our proposed *Recurrent Convolutional Unit* (RCU) is given in Figure 7.1(c). The input at any time instant t passes through 3D spatial convolution (with kernel of size $1 \times d \times d$, denoted by w_{xh}). The result is added to the output of a recurrent convolution operation, with kernel denoted by w_{hh}, of size $1 \times 1 \times 1$.
The result is termed the *hidden state* h_t of the unit. Analytically, a recurrent convolutional unit can be described by the following relation:

$$h(t) = (h_{t-1} * w_{hh} + x_t * w_{xh})/2.0, \tag{7.1}$$

where w_{hh} and w_{xh} are parameters of the RCU, and $*$ denotes the convolution operator. An RCU can be seen in the context of other modules of a network in Figure 7.2.

In practise, we replace every 3D convolution in Table 7.1 with RCU to obtain an RCN. In general, one could skip/add a few replacements to obtain a mixed network.

7.4.2 Unrolling a recurrent convolutional network

Figure 7.2 represents a simple recurrent convolutional network (RCN) composed by a single RCU unit, unrolled up to time t. At each time step t, an input x_t is processed by the RCU and the other layers to produce an output y_t.

The unrolling principle allows us to build an RCN from 2D/3D networks, e.g. by replacing 3D convolutions with RCUs in any I3D network. Indeed, the network architecture of our proposed model builds on the I3D network architecture shown in Table 7.1, where the same parameters (d, number of kernels) used for 3D convolutions are used for our RCU. Unlike I3D, however, our RCN does not require a temporal convolution size n (cfr. Table 7.1) as a parameter. As in 2D or I3D ResNet models [19, 148, 149], our proposed RCN also has residual connections. The hidden state h_t at time t is considered to be the output at that time instant – as such, it acts as input to next hidden state and to the whole next depth-level layer. Table 7.1 describes the network architecture of ResNet-18 [148] with 18 layers. Similarly, we can build upon other variants of ResNet.

7.4.3 Temporal resolution preservation

The output sizes for both I3D and our proposed RCN are shown in Table 7.1. Our RCN only uses spatial pooling and a convolutional layer for classification, unlike the spatiotemporal pooling of [17, 19, 57]. From Table 7.1, compared to I3D, RCN produces 16 classification score vectors with an input sequence length of 16.

This one-to-one mapping from input to output is essential in many tasks, ranging from temporal action segmentation [29, 84], to temporal action detection [20], to action tube detection [9]. In all such tasks, video-level accuracy is not enough, but we need frame-level results in terms, e.g., of detection bounding boxes and class scores. Temporal convolution behaves in a similar way to spatial convolution: it results in lower resolution feature maps as compared to the input as the depth of the network increases.

Unlike the temporal deconvolution proposed in [29,84], our RCN inherently addresses this problem (see Table 7.1). For a fair comparison, in our tests, we adjusted our baseline I3D model for dense prediction, by setting the temporal stride to 1 (§ 7.5.4). The resulting I3D model can produce dense predictions: given T frames as input, it will generate T predictions in 1-1 correspondence with the input frames.

7.4.4 Causality and long-term dependencies

A size-n temporal convolution operation uses a sequence $x_{t-n/2}, ..., x_t, ..., x_{t+n/2}$ as input to generate an output y_t at time t. In our Recurrent Convolutional Network, instead, y_t is a function of only the inputs $x_0, x_1, ..., x_t$ from the present and the past (up to the initial time step), as shown in Figure 7.2. Its independence from future inputs makes the output y_t at time t causal. Thus RCN, as presented here, is not only causal but poses no constraints on the modelling of temporal dependencies (as opposed to an upper bound of n in the case of temporal convolutions). Temporal dependencies are only limited by the input sequence length at training time.

As in traditional RNNs, we have the option to unroll the same network to model arbitrary input sequence lengths at test time, thus further increasing the horizon of temporal dependencies and show a substantial gain in performance. We will discuss it in more details while discussing the results in Section 7.5.7.

7.4.5 ImageNet initialisation for the 2D layers

The I3D model proposed by Carreira *et al.* [17] greatly owes its success to a good initialisation from 2D models trained on ImageNet [136]. By inflating these 2D models, we can benefit from their ImageNet pre-trained weights, as in most state-of-the-art 3D models [17, 18, 57]. We follow the same principle and initialise all 2D layers using the weights of available pre-trained 2D ResNet models [148]. It is noteworthy that the other state-of-the-art (2+1)D model by Tran *et al.* [19] cannot, instead, exploit ImageNet initialisation, because of the change in the number of kernels.

7.4.6 Identity Initialisation for the Hidden Layers

The presence of a hidden state convolution (w_{hh}, see Figure 7.2) layer at every depth level of the unrolled network makes initialisation a tricky issue. The random initialisation of the hidden state convolution component could destabilise the norm of the feature space between two 2D layers. In response to a similar issue, Le *et al.* [154] presented a simple way to initialise RNNs when used with ReLU [155] activation functions. Most state-of-the-art 2D models [138, 148] make indeed use of ReLU as activation function of choice for fast and optimal convergence [155].

Following the example of Le *et al.* [154] and others [156, 157], we initialise the weights of the hidden state convolution kernel (w_{hh}) with the identity matrix. Identity matrix initialisation is shown [154, 156] to capture longer term

dependencies. It also helps induce forgetting capabilities in recurrent models, unlike traditional RNNs.

7.5 Experiments

In this section, we evaluate our RCN on the challenging Kinetics [3] and MultiThumos [5] datasets to **answer the following questions**: i) How does our training setup, which uses 4 GPUs, compare with the 64 GPU training setup of [53] (Section 7.5.2)? ii) How do recurrent convolutions compare against 3D convolutions in the action recognition problem (§ 7.5.3)? iii) How does our RCN help solving the dense prediction task associated with action detection (§ 7.5.4)? Finally, iv) we validate our claims on the temporal causality and flexibility of RCN, and check whether those features help with longer-term temporal reasoning (§ 7.5.5).

Evaluation: For fair comparison, we computed clip-level and video-level accuracy as described in [19, 57]. Ten regularly sampled clips were evaluated per video, and scores were averaged for video-level classification. On videos of arbitrary length, we averaged all the predictions made by the unrolled versions of both our RCN and of I3D. Please refer to Section 2.9 to gather more details about the datasets and evaluation meterics (accuracy and mAP) used in this chapter.

7.5.1 Implementation details

In all our experiments we used *sequences of RGB* frames as input for simplicity and computational reasons. We used a batch size of 64 when training ResNet18-based models and 32 for models based on ResNet-34 and -50. The initial learning rate was set to 0.01 for batches of 64, and to 0.005 for batches of 32. We reduced the learning rate by a factor of 10 after both $250K$ and $350K$ iterations. Moreover, training was stopped after $400K$ iterations (number of batches). We used standard data augmentation techniques, such as random crop, horizontal flip with 50% probability, and temporal jittering.

7.5.2 Fair training setup

GPU memory consumption plays a crucial role in the design of neural network architectures. In our training processes, a maximum of 4 GPUs was used. Given our GPU memory and computational constraints, we only considered training

Table 7.2: Clip-level and video-level action recognition accuracy on the validation set of the Kinetics dataset for different ResNet18 based models, trained using 8-frame-long clips as input.

Network	#Params	Initialisation	Clip %	Video %
I3D [149]	33.4M	random	–	54.2
I3D [19]	33.4M	random	49.4	61.8
(2+1)D [19]	33.3M	random	52.8	64.8
I3D†	33.4M	random	49.7	62.3
RCN [ours]†	**12.8**M	random	51.0	63.8
(2+1)D†	33.4M	random	51.9	64.8
I3D†	33.4M	ImageNet	51.6	64.4
RCN [ours]†	**12.8**M	ImageNet	**53.4**	**65.6**

† trained with our implementation and training setup.

networks with 8-frame long input clips, except for ResNet34 which was trained with 16 frames long clips.

As explained above, we have GPU memory constrained, we will report results of previous basic 3D models [17, 19] re-implemented and trained by using the same amount of resources as our RCN. The main hyperparameters involved in the training of a 3D network are learning rate, batch size, and the number of iterations. These parameters are interdependent, and their optimal setting depends on the computational power at disposal. For instance, Tran *et al.* [19] would use 64 GPUs, with the training process distributed across multiple machines. In such a case, when vast computational resources are available [17,19,120], training takes 10-15 hours [19], allowing for time to identify the optimal parameters. The availability of such computational power, however, is scarce.

In a bid to reproduce the training setup of [19] on 4 GPUs, we re-implemented the I3D and (2+1)D models using ResNet18 and ResNet34 as a backbone. The ResNet18-I3D architecture is described in Table 7.1. Based on the latter, we built a (2+1)D [19] architecture in which we matched the number of parameters of separated convolutions to that of standard 3D convolutions, as explained in [19].

The results of the I3D and (2+1)D implementations reported in Tran *et al.* [19] are shown in the top half of Table 7.2. When comparing them with our implementations of the same networks in the bottom half, it is clear that our training is as performing as that of Tran *et al.* [19]. This allows a fair comparison of our results.

Why smaller clips as input: training a ResNet18-based model on Kinetics with 8 frame clips as input takes up to 2-3 days on 4 GPUs. Training a

Table 7.3: Video-level action classification accuracy of different models on the validation set of the Kinetics dataset.

Model	Clip-length	Initialisation	Acc%
ResNet34-(2+1)D [19]†	16	random	67.8
ResNet34-I3D [17] †	16	ImageNet	68.2
ResNet34-RCN [ours]⋆†	16	ImageNet	**70.3**
ResNet50-I3D [17] †	8	ImageNet	70.0
ResNet50-RCN [ours]⋆†	8	ImageNet	71.2
ResNet50-RCN-unrolled [ours]⋆†	8	ImageNet	**72.1**

⋆ causal model; † trained with our implementation and training setup.

ResNet50-based model takes up to 4-5 days. In principle, one could train the same model for longer input clip sizes, but the amount of GPU memory and time required to train would grow linearly. As an estimate, it would take more than two weeks to train a ResNet50 model on 64 long frame clips, assuming that all the hyperparameters are known (i.e., batch size, learning rate, step size for learning rate drop, and whether batch normalisation layers should be frozen or not).

For these reasons we stick to smaller input clips to train our models in a fair comparison setting, using the hyperparameter values from § 7.5.1.

7.5.3 Results on action recognition

We compared our RCN with both I3D and (2+1)D models in the action recognition problem on the Kinetics dataset. A fair comparison is shown in Table 7.2 with ResNet18 as backbone architecture. Table 7.3 shows the results with ResNet34 and ResNet50 as backbone, trained on 16 frame and 8 frame clips, respectively. It is clear from these figures that RCN significantly outperforms state-of-the-art 3D networks – e.g. our network outperforms the equivalent I3D network by more than 2% across the board.

The ability to model **long-term temporal reasoning** of RCN is attested by the performance of the unrolled version (last row of Table 7.3). It shows that, even though the network is trained on input clip of 8 frames, it can reason over longer temporal horizons at test time. The corresponding unrolled I3D version (the last classification layer is also convolutional, see Table 7.1) showed no substantial improvement in performance – in fact, a slight drop.

Comparison with I3D variants: the main variants of the I3D model are separated 3D convolutions with gating (S3Dg) [18] and with non-local operators (NL) [57]. We think it appropriate to take a closer look at these variants of

Table 7.4: Action detection/segmentation results on MultiThumos dataset, mAP computed from dense prediction at every frame (mAP@1) and every 8th frame (mAP@8).

Network	Input	mAP@1 %	mAP@8 %
Two-stream+LSTM [5]⋆	RGB+FLOW	28.1	-
MultiLSTM [5]⋆	RGB+FLOW	29.7	-
Inception-I3D by [83]	RGB+FLOW	-	30.1
Inception-I3D + SE [83]	RGB+FLOW	-	36.2
ResNet50-I3D [baseline]	RGB	34.8	36.9
ResNet50-RCN [ours]⋆	RGB	35.3	37.3
ResNet50-RCN-unrolled [ours]⋆	RGB	**36.2**	**38.3**

⋆ causal model

I3D as they provide state-of-the-art performance, albeit being all anti-causal. In [18, 57] the application of non-local or gating operations to I3D yields the best performances to date, mainly thanks to training on longer clips given a large amount of GPU memory at their disposal (S3Dg [18] models are trained using 56 GPUs, [57] uses 8 GPUs with 16GB memory each). The best version of I3D-NL achieves an accuracy of 77.7%, but uses 128 frames and ResNet101 as backbone network; hence we do not deem fair to compare it with our models (which only use 8 frame long clips). It would take almost a month to train such a network using 4 GPUs. What needs to be stressed is that *gating and NL* operations are not at all constrained to be applied on top of I3D or S3D models: indeed, they *can also be used in conjunction with (2+1)D and our own RCN model*. As in this work we focus on comparing our network with other 3D models, we chose I3D and (2+1)D as baselines (Sec. 7.5.3).

Training on longer sequences: we tried training on longer sequences (**32** frames) by reducing the batch size to 8 with Resnet50 as base network. Despite a sub-optimal training procedure, RCN was observed to still outperform I3D by a margin of 1.5%. A closer to optimal training procedure with Resnet50 (as in [19, 57]), is very likely to yield even better results.

7.5.4 Results on temporal action detection

We also evaluate our model on the temporal action detection problem on the MultiThumos [5] dataset. The latter is a dense label prediction task. As a baseline, we use a temporal resolution preserving a version of I3D introduced in Section 7.4.3. ResNet50 is employed as a backbone for both our RCN and the baseline I3D. To capture the longer duration, we use 16 frame clips as input; the sampling period is 4 frames. Both networks are initialised with the respective

Table 7.5: Comparison between RCN and other causal models on the validation set of Mini-Kinetics [120] (First half of the table) and Kinetics-400 (last half of the table). ResNet-50 (R50) is used as base model.

Model	Clip-length	Acc%
InceptionV1-I3D [120]	64	71.8
InceptionV1-I3D-seq [120]*	64	71.1
InceptionV1-I3D-par [120]*	64	54.5
R50 - I3D†	8	70.0
R50 - RCN [ours]*†	8	71.2
R50 - RCN - urolled [ours]*†	8	72.2

* causal model, unlike respective I3D version
† trained with our implementation and training setup.

models pretrained on Kinetics. The initial learning rate is set to 0.001 and dropped after $14K$ iterations, a batch size of 16 is used, and trained up to $20K$ iterations. Similar to [83], we use binary cross-entropy as loss function.

We use the evaluation setup of [5] and [83], and computed both frame-wise mean Average Precision at 1 (mAP@1) (in which case a prediction needs to be made for each frame) and [83] mAP@8 (every 8th frame).

Table 7.4 shows the performance of our models along with that of other state-of-the-art methods. Two LSTM-based causal models presented by [5] are shown in rows 1 and 2. Piergiovanni *et al.* [83] use pretrained I3D [17] to compute features, but do not train I3D end-to-end, hence their performance is lower than in our version of I3D. Our RCN outperforms all other methods, including anti-causal I3D+Super-Events (SE) [83] and the I3D baseline. It is safe to say that RCN is well applicable to dense prediction tasks as well.

7.5.5 Causality and temporal reasoning

A comparison with other causal methods is a must, as we claim the causal nature of the network to be the main contributions of our work, making RCN best suited to online applications such as action detection and prediction. In Section 7.5.4 we have already shown that our model excels in the task of temporal action detection.

Carreira *et al.* [120] proposed two causal variants of the I3D network. Their sequential version of I3D, however, shows a slight drop [120] in performance as compared to I3D, as seen in the first and second row of Table 7.5. Their parallel version is much faster than the sequential one but suffers from an even more significant performance decline 71.8% to 54.5% [120], as seen in the first and third row of Table 7.5.

In contrast, our causal/online model not only outperforms other causal mod-

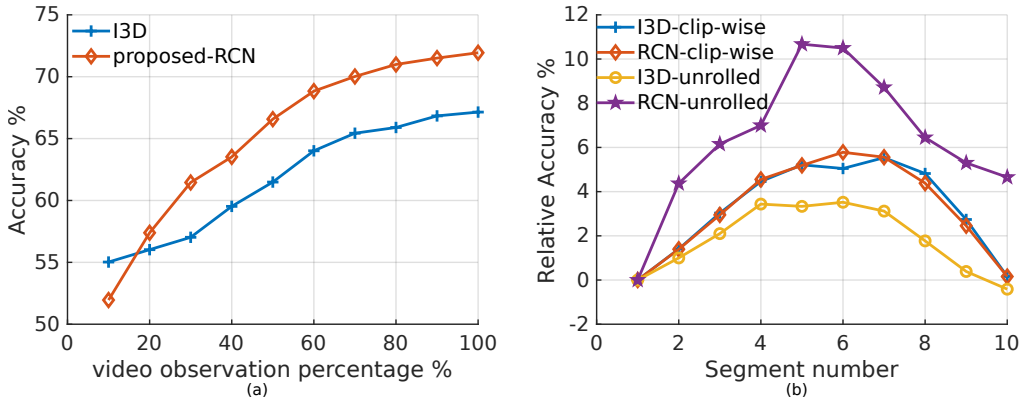


Figure 7.3: (a) Online/early action prediction task: accumulated scores are used to compute the accuracy against the label of the entire video as a function of the video observation percentage. (b) relative (difference from first segment) accuracy of each of 10 regularly sampled segments.

els (see Table 7.4) but beats as well strong, inherently anti-causal state-of-the-art 3D networks on a large scale dataset such as Kinetics (see last half of Table 7.5).

In addition, as in [9, 104], we can use the **early action prediction task** to evaluate the sequential temporal reasoning of RCN. The task consists in guessing the label of an entire action instance (or video, if containing a single action) after observing just a fraction of video frames. Accumulated output scores up to time t are used to predict the label of the entire video. The idea is that a system should improve its classification accuracy as it observes a larger fraction of the input video, by exploiting temporal context. Figure 7.3(a) shows that our RCN improves drastically as more video frames are observed when compared to I3D. It indicates that RCN has superior anticipation ability, albeit starting slowly in first 10% of the video.

Furthermore, to provide useful cues about causality and temporal reasoning, we designed an original **segment-level classification** evaluation setting. Namely, the outputs of the models being tested are divided into ten regularly sampled segments and the difference between the accuracy for each segment and that for the first segment is computed, as shown in Figure 7.3(b). Within this setting, we compared the I3D baseline with RCN in two different modalities, one considering clip-wise outputs in a sliding window fashion, the other obtained by unrolling both I3D and RCN over test videos of arbitrary length.

Notably, middle segments provide the best relative improvement, which is reasonable as it indicates that the middle part of the video is the most informative. Secondly, the last segment (no. 10) has the lowest relative accuracy of all, except for RCN-unrolled. The relative accuracy of a pure causal system,

though, should improve monotonically, i.e., exploit all it has seen. Instead, all compared models end up at the same performance they started with, except for unrolled RCN for which the final accuracy is almost 5% higher than the initial one. We can conclude that unrolled RCN has a longer-term memory than unrolled I3D or both sliding window-based I3D/RCN.

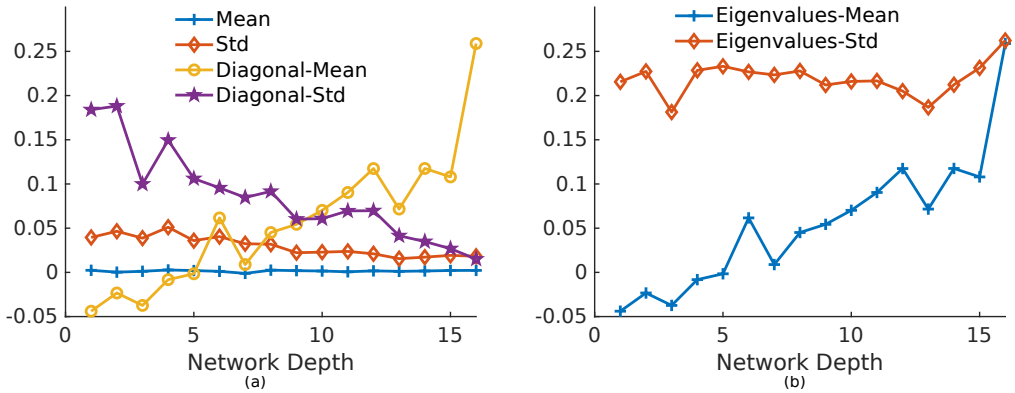


Figure 7.4: (a) Mean and standard deviation (Std) of all the entries of the weight matrices (w_{hh}) of the hidden state at every RCU layer of RCN, as well as those of just the diagonal elements. (b) Mean and Std of the eigenvalues of the hidden state weight matrices at every RCU layer of a 18-layer RCN.

7.5.6 Evolution of recurrence with network depth

It is another aspect that can provide clues about RCN's temporal flexibility. To this purpose, we examine the statistics of the weight matrices (w_{hh}) associated with the hidden state at every RCU layer in the RCN network. In Figure 7.4(a) we can see that the mean of the diagonal elements of the weight matrices increases and their standard deviation decreases with the depth of the network. This means that the w_{hh} matrix becomes sparser as network depth grows. In our view, this phenomenon is associated with RCN putting more focus on feature learning in the early part of the network, and emphasising temporal reasoning at later depths as the temporal reasoning horizon ('receptive field') increases. In other words, RCN learns to select the layers which should contribute towards temporal reasoning automatically. A similar phenomenon is observed in networks based on temporal convolutions [18].

Arjovsky [157] argue that if the eigenvalues of the hidden state-to-hidden state weight matrix diverge from the value 1, optimisation becomes difficult due to the vanishing gradient problem. Chang *et al.* [158] explore a similar idea. Taking an ordinary differential equation view of RNNs, they argue that their stability for long-term memory is related to the eigenvalues of the weight matrices. Figure 7.4(b) shows that in RCN the mean eigenvalue does rise to-

wards 1 as network depth increases, suggesting that later layers are more stable in terms of long-term memory whereas earlier layers are not concerned with long-term reasoning.

Table 7.6: Video-level and clip-level action recognition accuracy on the Kinetics validation set for different initialisation of 2D layer (w_{xh}) and hidden state unit weights (w_{hh}) in RCU with ResNet-18-based RCN models trained on 8 frame-long clip as an input.

w_{xh} init	w_{hh} init	Clip-Acc%	Video-Acc%
Random	Random	49.2	61.2
Random	Identity	49.7	62.1
ImageNet	Random	50.7	62.9
ImageNet	Identity	**53.4**	**65.6**

7.5.7 Effect of weight initialisation.

Weights of inflated 2D layers in ResNet (base networks) [148] based I3D and our RCN networks are initialised with weights from a pre-trained ImageNet model train on RGB images. In the case of RCN inflation in the third dimension is 1, practically, it is 2D weight matrix with the third dimension being 1. As a result, we do need to replicate the weights like in the 3D layer of I3D.

Random initialisation for hidden state parameters resulted in sub-optimal training. Thus, in all the experiments with RCN, we used identity matrix initialisation instead. Here we show the result of training RCN with different initialisation of hidden-state w_{hh} convolution and spatial w_{xh} convolution in Table 7.6. The first row of the table where both w_{xh} and w_{hh} of all the RCUs in RCN is initialised randomly using normal initialisation process described in [159].

Table 7.6 show the result of different initialisation of w_{xh} and w_{hh} respectively. It is clear from the last row of the table that RCN performs best if w_{xh} and w_{hh} are initialised with ImageNet and identity matrix respectively. It is noteworthy that the performance difference is relatively small in first three rows of that table as compared to the last row jump.

7.5.8 Discussion

In the light of RCN's superior results on temporal action detection (§ 7.5.4), early action prediction (see Figure 7.3(a)), long-term temporal reasoning (in its unrolled incarnation) at segment-level and for action recognition (§ 7.5.3, see the last row of Table 7.3), it is fair to say that the proposed Recurrent Convolutional Network is the best performing causal network out there.

Why layer-wise design: we tried replacing 3D CONV by RCU, (i) only in the last four layers and (ii) on four regularly sampled layers of Resnet50. This led to lower performance (69% and 68% respectively), compared to 71% when RCU replaces all 18 3D CONVs. This is consistent with previous approaches [18, 57], we can conclude that replacing every 3D conv with RCU leads to the best performance.

The number of parameters in our proposed RCN model is 12.8 million (M), as opposed to 33.4M in both the I3D and (2+1)D models, see Table 7.2. It is remarkable to see that, despite a 2.6 times reduction in the number of parameters, RCN still outperforms both I3D and (2+1)D when trained using ImageNet initialisation. Further, RCN surpasses I3D also under random initialisation, while using 2.6 times fewer model parameters. We measured the floating-point operations (FLOPs) for I3D, R(2+1)D, and RCN, recording 41MMac, 120MMac, and 54MMac, respectively. Thus, RCN requires half the FLOPs as compared to (2+1)D, and is comparable to I3D because RCN preserves temporal resolution. We computed the average time taken to process ten-second long videos of the Kinetics dataset. This takes 0.4s, 0.8s, and 0.9s for I3D, RCN, and (2+1)D respectively.

ImageNet initialisation proves to be useful for both the I3D and our RCN models. While (2+1)D performs (Table 7.2, row 6) better than RCN (row 5) with the random initialisation, our RCN recovers to improve over (2+1)D (row 6) with ImageNet initialisation, whereas (2+1)D cannot make use of free ImageNet initialisation. This seems to be a severe drawback for the (2+1)D model, and a big advantage for I3D and RCN. One may argue that, if the purpose is to build on existing 2D models, then RCN and I3D are a better choice, whereas if new 3D models are preferred then (2+1)D might prove useful. The latter does provide better performance with the random initialisation, but at the price of requiring many more parameters than RCN.

Random initialisation for hidden state parameters resulted in sub-optimal results. Thus, in all the experiments with RCN, we used identity matrix initialisation instead. Identity matrix initialisation helps to capture forget capabilities as well, as suggested by [154]. An ablation study on the effect of initialisation is provided in Section 7.5.7.

On input clip length: input clip length is another factor in determining the final performance of any network. We can see from the Table that all the

Inception-based models are trained on 64 frame-long clips, one of the reasons why Inception nets work better that ResNet models while using fewer parameters. Among the latter, ResNet101-I3D-NL [57] is shown to work better with an even longer input clip length. Thus, the evidence supports that training on larger input sequences boosts performance while clashing with memory limitations.

In our experiments, as mentioned, we stuck to 8 or 16 frame clips as input and compared our proposed RCN with baseline I3D models. We think this provides enough evidence of the validity of our proposal to move away from temporal convolutional networks [17–19, 57], and replace them with more sophisticated *and* causal structures. As with the role of additional layers, it is fair to predict that more extensive training on more extended clips (32,64,128) has a serious potential to take RCN to outperform state of the art in absolute terms.

On efficient training of 3D networks: two basic things are clear from our experience with training heavy 3D models (I3D, (2+1)D, RCN) on large-scale datasets such as Kinetics. Firstly, training is very computationally expensive and memory bulky; secondly, longer input clips are crucial to achieving better optimisation which, however, renders the first issue even more severe. We feel that how to train these model efficiently is still a wide-open problem, whose solution is essential to speed up the process for broader adoption. We observed that ImageNet initialisation does speed up the training procedure, and helps reach local minima much quicker. In the case of both I3D and RCN, ImageNet initialisation improves the video classification accuracy on Kinetics by almost 3% compared to random initialisation when using the same number of training iterations, as shown in the first and last row of Table 7.6.

The bottom line is that we should strive for more efficient implementations of 3D models for the sake of their adoption.

7.6 Summary and limitations

Summary: In this work, we presented a recurrence-based convolutional network (RCN) able to generate causal spatiotemporal representations by converting 3D CNNs in to causal 3D CNNs while using 2.6 times fewer parameters compared to its traditional 3D counterparts. RCN can model long-term temporal dependencies without the need to specify temporal extents. The proposed RCN is not only causal in nature and temporal resolution-preserving but was also shown to outperform the main baseline 3D networks in all the fair compar-

isons we ran. We showed that ImageNet-based initialisation is at the heart of the success of 3D CNNs. Indeed, although RCN is recurrent in nature, it can still utilise the weights of a pre-trained 2D network for initialisation.

Limitations: our proposed network has all the nice properties like causality, flexible temporal reasoning temporal resolution preservation. However, there some limitations as follows: i) it is only evaluated on small input clip lengths of 8-16 frames, ii) it require more GPU memory to train than a 3D network because of temporal resolution preservation iii) it relies on good initialisation for its performance. Despite these problems, our RCN is the best causal video representation model out there.

Looking ahead: The causal nature of our recurrent 3D convolutional network opens up manifold research directions, with direct and promising potential application in areas such as online action detection and future event/action prediction. In the next chapter, we will discuss the potential application of causal 3D CNNs along with the inspiration from previous chapters.

Chapter 8

Conclusions and Future Research Directions

In this thesis , we have introduced a number of novel deep learning based frameworks (Chapter 3, 4, 5, 5 & 7) to solve for variety video understanding problems. In particular, we addressed the problem of action detection, online and real-time action detection, early action prediction, future action tube prediction and causal video representations. All these problems have applications in a real-world scenario, e.g. online and real-time action detection system could be used in real-time video surveillance, and future action tube prediction has applications in self-driving and many others.

To conclude this thesis, we divide this chapter into two sections. First, we summarise (§ 8.1) the contributions of the presented work in this thesis in chapter-wise order. Lastly, we will conclude by introducing some the most prominent direction for the future works(§ 8.1).

8.1 Summary of contributions of the thesis

8.1.1 Action detection using frame-level deep features

We presented a novel spatiotemporal action detection pipeline and temporal labelling framework, where we show that supervised proposal is important for predicting frame-level detection boxes and the associated action class scores. We propose an original fusion strategy for merging appearance and motion cues based on the softmax probability scores and spatial overlaps of the detection of bounding boxes. We propose to solve action tube construction by two different optimisation formulation. Finally, we proposed an efficient dynamic programming based solution to temporal localisation of action in tubes and videos. As a result, our pipeline was able to outperform the previous state-of-the-art methods

in action detection.

8.1.2 Online and real-time action detection

We will describe the first online and real-time action detection system which can detect multiple co-occurring actions in untrimmed videos. We devise an original, greedy online algorithm capable of generating multiple action tubes incrementally, which also provides early action class label predictions. Further, we extend the online action detection task to untrimmed videos for the first time. We show that such a system can work in an online and real-time fashion and also outperform previous state-of-the-art action detection methods.

8.1.3 Flexible micro-tube proposals

Moving away from the frame-level detector and tube construction methods in previous chapters, we propose two novel deep networks to incorporate information from multiple frames. Flexible anchor proposal method is proposed with the help of a hidden Markov model (HMM) formulation, which can be approximated efficiently into a transition matrix. We show that such a network can handle significant spatial movement in dynamic actors without penalising more static actions. Further, we observed that it is a scalable solution for training models on both sparse and dense annotations.

8.1.4 Predicting Future locations of action tubes

We showed that a micro-tube prediction network can be extended to predict the future of each action tube individually. Thanks to online tube construction method proposed previous chapters, we propose a future action tube prediction framework. Here, we linked that the online nature of methods is essential to future prediction. We demonstrate that training a network to make predictions also helps in improving action detection performance. We also demonstrate that feature-based fusion works better than late fusion in the context of spatiotemporal action detection, unlike late fusion in previous chapters.

8.1.5 Causal 3D representations

Motivated by the applications of online methods, we propose to convert temporal-convolutions based state-of-the-art anti-causal 3D CNNs to causal 3D CNNs by an original convolutional network with recurrent hidden states at each depth level. We show that it can directly benefit from model initialisation via ImageNet pre-trained weights, as opposed to state of the art approaches, and in

line with clear emerging trends in the field. It also allows flexible temporal reasoning, as it exploits information coming from all the input sequence observed.

8.2 Overall conclusion

We presented various action detection methods in this thesis, the main thread has been online processing of videos. We see that online processing not only help action detector to be applicable in real-world applications but also it can help such online detection to predict action early or predict future of action tubes. We believe that such online processing would result in much more applications. From another main contribution point of view, we show that we can take care of the dynamic nature of action while designing the reference anchor proposals for an action detector, such a detector can be scaled to handle both sparse and dense annotations. Finally, we show that we can convert anti-causal 3D CNNs into causal 3DCNNs with the help of recurrence, which leads to flexible temporal reasoning.

8.3 Future research directions

In this section, we bring forward some prominent research directions for future work based on the experimental results reported in this thesis and the very latest advancements in computer vision and machine learning.

8.3.1 Causal spatiotemporal action detection with 3D CNNs

We have discussed in Chapter 2 in Section 2.4.3 that 3D CNNs enjoy superior performance in action detection task due to their high representational power. However, all such methods are anti-causal. We can use causal RCN proposed in Chapter 7 for better online action detector, also an extension to future anticipation tasks. RCN can also preserve the temporal resolution, which would make its application in streaming applications easier. All the existing methods using 3D CNNs for action detection use a 2D CNNs in parallel on keyframes for bounding box detection, then they pool classification features from 3DCNNs for that box. It is still to be tested if RCN would need such a 2D detector in parallel or not. Also, it yet to be seen if we can convert latest slow-fast 3D CNN [58] into a causal 3D CNN.

8.3.2 Improving action representation

More recent works on 3D representation [18,19] has shown that factorised convolutions not only reduce the number of parameters in a network but also improve the performance. Subsequently, Feichtenhofer *et al.* [58], show the two parallel streams of a network can process information and different rate and help gain better performance on action recognition tasks. It would be interesting to see if these kinds of networks can be made causal. One of the ways to make these networks causal is to pre-trained these networks the replace 3D convolutions by recurrent convolutions. Also, it would be interesting to try to train recurrent convolutions from scratch, as 3D CNNs are trained from scratch in [19, 58].

8.3.3 Improving temporal action detection

We observed in Chapter 2 in Section 2.2 that proposal based methods for temporal action detection are best at finding temporal boundaries of action instance. It would be interesting to see if action paths (Figure. 3.3) can be trimmed with temporal boundary prediction network [80]. We can see from Table 4.3 that the classes with low ratio of action instance duration over video duration have a very low performance, i.e. occupy the less temporal extent in the video. It means these classes needed better temporal detection, e.g. 'Cricket bowling' or 'Volleyball Spiking'. We feel that the adaptation of temporal detection literature into spatiotemporal action detection method has a lot of room to improve.

References

[1] K. Soomro, A. Zamir, and M. Shah, "Ucf101: a dataset of 101 human action classes from videos in the wild (2012)," *arXiv preprint arXiv:1212.0402*.

[2] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

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

[4] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, "Hollywood in homes: Crowdsourcing data collection for activity understanding," in *Proceedings of the European Confercence Computer Vision (ECCV))*, 2016.

[5] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei, "Every moment counts: Dense detailed labeling of actions in complex videos," *International Journal of Computer Vision*, vol. 126, no. 2-4, pp. 375–389, 2018.

[6] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[7] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar, "THUMOS challenge: Action recognition with a large number of classes." `http://crcv.ucf.edu/THUMOS14/`, 2014.

[8] G. Gkioxari and J. Malik, "Finding action tubes," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[9] G. Singh, S. Saha, M. Sapienza, P. Torr, and F. Cuzzolin, "Online real-time multiple spatiotemporal action localisation and prediction," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, pp. 3637–3646, 2017.

[10] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black, "Towards understanding action recognition," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, pp. 3192–3199, IEEE, 2013.

[11] C. Gu, C. Sun, S. Vijayanarasimhan, C. Pantofaru, D. A. Ross, G. Toderici, Y. Li, S. Ricco, R. Sukthankar, C. Schmid, *et al.*, "Ava: A video dataset of spatio-temporally localized atomic visual actions," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] C. Wolf, J. Mille, E. Lombardi, O. Celiktutan, M. Jiu, M. Baccouche, E. Dellandra, C.-E. Bichot, C. Garcia, and B. Sankur, "The LIRIS Human activities dataset and the ICPR 2012 human activities recognition and localization competition," Tech. Rep. RR-LIRIS-2012-004, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon, Mar. 2012.

[13] P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Learning to track for spatio-temporal action localization," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3164–3172, 2015.

[14] S. Saha, G. Singh, M. Sapienza, P. H. S. Torr, and F. Cuzzolin, "Deep learning for detecting multiple space-time action tubes in videos," in *Proceedings of the British Machine Vision Conference*, 2016.

[15] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid, "Action tubelet detector for spatio-temporal action localization," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, 2017.

[16] R. Hou, C. Chen, and M. Shah, "Tube convolutional neural network (t-cnn) for action detection in videos," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, 2017.

[17] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4724–4733, 2017.

[18] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, "Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification," in *Proceedings of the European Conference Computer Vision (ECCV)*, pp. 305–321, 2018.

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

[20] G. Singh and F. Cuzzolin, "Untrimmed video classification for activity detection: submission to activitynet challenge," *arXiv preprint arXiv:1607.01979*, 2016.

[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Proceedings of the European Confernce Computer Vision (ECCV)*, pp. 21–37, Springer, 2016.

[22] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Proceedings of the European Conference Computer Vision (ECCV)*, pp. 25–36, Springer, 2004.

[23] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool, "Fast optical flow using dense inverse search," in *Proceedings of the European Conference on Computer Vision*, pp. 471–488, Springer, 2016.

[24] S. Saha, G. Singh, and F. Cuzzolin, "Amtnet: Action-micro-tube regression by end-to-end trainable deep architecture," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, 2017.

[25] G. Singh, S. Saha, and F. Cuzzolin, "TraMNet - Transition Matrix Network for high-performance spatio-temporal action localisation," in *Proceedings of the Asian Confernce Computer Vision (ACCV)*, 2018.

[26] G. Singh, S. Saha, and F. Cuzzolin, "Predicting action tubes," in *Proceedings of the European Confercence Computer Vision Workshop (ECCVW) on Anticipating Human Behavior*, 2018.

[27] G. Singh and F. Cuzzolin, "Recurrent convolution for causal 3d cnns," in *Proceedings of the International Conference on Computer Vision Workshops (ICCVW)*, 2019.

[28] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–970, 2015.

[29] C. L. M. D. F. René and V. A. R. G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, 2017.

[30] A. F. Bobick and J. W. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 3, pp. 257–267, 2001.

[31] Z. Zhang, Y. Hu, S. Chan, and L.-T. Chia, "Motion context: A new representation for human action recognition," in *European Conference on Computer Vision*, pp. 817–829, Springer, 2008.

[32] I. Laptev, "On space-time interest points," *International journal of computer vision*, vol. 64, no. 2-3, pp. 107–123, 2005.

[33] I. Laptev and P. Pérez, "Retrieving actions in movies," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, pp. 1–8, IEEE, 2007.

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

[35] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 886–893, IEEE, 2005.

[36] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision (IJCV)*, 2004.

[37] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Proceedings of the European Conference Computer Vision (ECCV)*, pp. 428–441, Springer, 2006.

[38] A. Klaser, M. Marszałek, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 275–1, British Machine Vision Association, 2008.

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

[40] H. Wang and C. Schmid, "Action Recognition with Improved Trajectories," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, pp. 3551–3558, 2013.

[41] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision (ECCV)*, vol. 1, pp. 1–2, Prague, 2004.

[42] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, IEEE, 2007.

[43] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[44] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1097–1105, 2012.

[46] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 568–576, 2014.

[47] C. Feichtenhofer, 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*, pp. 1933–1941, 2016.

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

[49] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action

recognition," in *European Conference on Computer Vision (ECCV)*, pp. 20–36, Springer, 2016.

[50] S. Ma, L. Sigal, and S. Sclaroff, "Learning activity progression in lstms for activity detection and early detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1942–1950, 2016.

[51] G. A. Sigurdsson, S. K. Divvala, A. Farhadi, and A. Gupta, "Asynchronous temporal fields for action recognition.," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 5, p. 7, 2017.

[52] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[53] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, 2015.

[54] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi, "Human action recognition using factorized spatio-temporal convolutional networks," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, pp. 4597–4605, 2015.

[55] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, pp. 5534–5542, IEEE, 2017.

[56] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1510–1517, 2018.

[57] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[58] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, 2019.

[59] A. Diba, M. Fayyaz, V. Sharma, M. Paluri, J. Gall, R. Stiefelhagen, and L. V. Gool, "Holistic large scale video understanding," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, 2019.

[60] A. Gaidon, Z. Harchaoui, and C. Schmid, "Temporal localization of actions with actoms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 11, pp. 2782–2795, 2013.

[61] Y. Tian, R. Sukthankar, and M. Shah, "Spatiotemporal deformable part models for action detection," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2642–2649, IEEE, 2013.

[62] D. Oneata, J. Verbeek, and C. Schmid, "Efficient action localization with approximately normalized fisher vectors," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2545–2552, 2014.

[63] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[64] Z. Shou, D. Wang, and S.-F. Chang, "Temporal action localization in untrimmed videos via multi-stage cnns," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1049–1058, 2016.

[65] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[66] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[67] R. De Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, and T. Tuytelaars, "Online action detection," *arXiv preprint arXiv:1604.06506*, 2016.

[68] B. Singh and M. Shao, "A multi-stream bi-directional recurrent neural network for fine-grained action detection," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[69] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei, "End-to-end learning of action detection from frame glimpses in videos," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[70] A. Montes, A. Salvador, S. Pascual, and X. Giro-i Nieto, "Temporal activity detection in untrimmed videos with recurrent neural networks," *arXiv preprint arXiv:1608.08128*, 2016.

[71] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 156–165, 2017.

[72] K. Kulkarni, G. Evangelidis, J. Cech, and R. Horaud, "Continuous action recognition based on sequence alignment," *International Journal of Computer Vision*, vol. 112, no. 1, pp. 90–114, 2015.

[73] G. Evangelidis, G. Singh, and R. Horaud, "Continuous gesture recognition from articulated poses," in *CharlearnLAP ECCV Workshops*, 2014.

[74] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem, "Daps: Deep action proposals for action understanding," in *European Conference on Computer Vision*, pp. 768–784, Springer, 2016.

[75] J. Gao, Z. Yang, K. Chen, C. Sun, and R. Nevatia, "Turn tap: Temporal unit regression network for temporal action proposals," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3628–3636, 2017.

[76] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin, "Temporal action detection with structured segment networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2914–2923, 2017.

[77] H. Xu, A. Das, and K. Saenko, "R-c3d: Region convolutional 3d network for temporal activity detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 5783–5792, 2017.

[78] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. Carlos Niebles, "Sst: Single-stream temporal action proposals," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2911–2920, 2017.

[79] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar, "Rethinking the faster r-cnn architecture for temporal action localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1130–1139, 2018.

[80] T. Lin, X. Zhao, H. Su, C. Wang, and M. Yang, "Bsn: Boundary sensitive network for temporal action proposal generation," in *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

[81] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 91–99, 2015.

[82] J. B. Roerdink and A. Meijster, "The watershed transform: Definitions, algorithms and parallelization strategies," *Fundamenta informaticae*, vol. 41, no. 1, 2, pp. 187–228, 2000.

[83] A. Piergiovanni and M. S. Ryoo, "Learning latent super-events to detect multiple activities in videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5304–5313, 2018.

[84] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang, "Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1417–1426, IEEE, 2017.

[85] A. Prest, V. Ferrari, and C. Schmid, "Explicit modeling of human-object interactions in realistic videos," *IEEE Transection on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 35, no. 4, pp. 835–848, 2013.

[86] T. Lan, Y. Wang, and G. Mori, "Discriminative figure-centric models for joint action localization and recognition," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, pp. 2003–2010, IEEE, 2011.

[87] A. Klaser, M. Marszałek, C. Schmid, and A. Zisserman, "Human focused action localization in video," in *SGA 2010-International Workshop on Sign, Gesture, and Activity, ECCV 2010 Workshops*, vol. 6553, pp. 219–233, Springer, 2010.

[88] L. Wang, Y. Qiao, and X. Tang, "Video action detection with relational dynamic-poselets," in *Proceedings of the European Conference Computer Vision (ECCV)*, pp. 565–580, Springer, 2014.

[89] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Proceedings of the European Conference Computer Vision (ECCV)*, pp. 391–405, Springer, 2014.

[90] X. Peng and C. Schmid, "Multi-region two-stream R-CNN for action detection," in *Proceedings of the European Confercence Computer Vision (ECCV)*, (Amsterdam, Netherlands), Oct. 2016.

[91] J. He, M. S. Ibrahim, Z. Deng, and G. Mori, "Generic tubelet proposals for action localization," *arXiv preprint arXiv:1705.10861*, 2017.

[92] L. Cao, Z. Liu, and T. S. Huang, "Cross-dataset action detection," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1998–2005, IEEE, 2010.

[93] M. Sapienza, F. Cuzzolin, and P. H. Torr, "Learning discriminative space–time action parts from weakly labelled videos," *International Journal of Computer Vision*, vol. 110, no. 1, pp. 30–47, 2014.

[94] M. Sapienza, *Recognising and localising human actions*. PhD thesis, Oxford Brookes University, 2014.

[95] D. Li, Z. Qiu, Q. Dai, T. Yao, and T. Mei, "Recurrent tubelet proposal and recognition networks for action detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 303–318, 2018.

[96] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[97] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krahenbuhl, and R. Girshick, "Long-term feature banks for detailed video understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 284–293, 2019.

[98] R. Girdhar, J. a. Carreira, C. Doersch, and A. Zisserman, "Video Action Transformer Network," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[99] K. Duarte, Y. S. Rawat, and M. Shah, "Videocapsulenet: A simplified network for action detection," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[100] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang, "Real-time action recognition with enhanced motion vector cnns," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[101] T.-H. Yu, T.-K. Kim, and R. Cipolla, "Real-time action recognition by spatiotemporal semantic and structural forests.," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2010.

[102] C. Schüldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3, pp. 32–36, IEEE, 2004.

[103] M. S. Ryoo and J. Aggarwal, "Ut-interaction dataset, icpr contest on semantic description of human activities (sdha)," in *IEEE International Conference on Pattern Recognition Workshops*, vol. 2, p. 4, 2010.

[104] M. Ryoo, "Human activity prediction: Early recognition of ongoing activities from streaming videos," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, pp. 1036–1043, IEEE, 2011.

[105] M. Hoai and F. De la Torre, "Max-margin early event detectors," *International Journal of Computer Vision*, vol. 107, no. 2, pp. 191–202, 2014.

[106] T. Lan, T.-C. Chen, and S. Savarese, "A hierarchical representation for future action prediction," in *Proceedings of the European Confercence Computer Vision (ECCV)*, pp. 689–704, Springer, 2014.

[107] M. S. Aliakbarian, F. S. Saleh, M. Salzmann, B. Fernando, L. Petersson, and L. Andersson, "Encouraging lstms to anticipate actions very early," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, vol. 1, 2017.

[108] K. Soomro, H. Idrees, and M. Shah, "Predicting the where and what of actors and actions through online action localization," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[109] H. S. Behl, M. Sapienza, G. Singh, S. Saha, F. Cuzzolin, and P. H. Torr, "Incremental tube construction for human action detection," in *Proceedings of the of the British Machine Vision Conference (BMVC)*, 2017.

[110] E. Nazerfard and D. J. Cook, "Using bayesian networks for daily activity prediction.," in *AAAI Workshop: Plan, Activity, and Intent Recognition*, 2013.

[111] Y. Kong, Z. Tao, and Y. Fu, "Deep sequential context networks for action prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1473–1481, 2017.

[112] A. Zunino, J. Cavazza, A. Koul, A. Cavallo, C. Becchio, and V. Murino, "Predicting human intentions from motion cues only: A 2d+ 3d fusion approach," in *Proceedings of the 2017 ACM on Multimedia Conference*, pp. 591–599, ACM, 2017.

[113] H. Koppula and A. Saxena, "Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 792–800, 2013.

[114] M. H. Tahmida Mahmud and A. K. Roy-Chowdhury, "Joint prediction of activity labels and starting times in untrimmed videos," in *Proceedings of the International Confernce on Computer Vision (ICCV)*, vol. 1, 2017.

[115] Z. Yang, J. Gao, and R. Nevatia, "Spatio-temporal action detection with cascade proposal and location anticipation," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[116] C. Vondrick, H. Pirsiavash, and A. Torralba, "Anticipating the future by watching unlabeled video," *arXiv preprint arXiv:1504.08023*, 2015.

[117] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–971, 2016.

[118] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *European Conference on Computer Vision*, pp. 201–214, Springer, 2012.

[119] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 336–345, 2017.

[120] J. Carreira, V. Patraucean, L. Mazare, A. Zisserman, and S. Osindero, "Massively parallel video networks," in *Proceedings of the European Conference Computer Vision (ECCV)*, 2018.

[121] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3074–3082, 2015.

[122] G. Yu and J. Yuan, "Fast action proposals for human action detection and search," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1302–1311, 2015.

[123] H. Idrees, A. R. Zamir, Y.-G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah, "The thumos challenge on action recognition for videos in the wild," *Computer Vision and Image Understanding*, vol. 155, pp. 1–23, 2017.

[124] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," in *ICCV*, pp. 2556–2563, IEEE, 2011.

[125] P. Weinzaepfel, X. Martin, and C. Schmid, "Human action localization with sparse spatial supervision," *arXiv preprint arXiv:1605.05197*, 2016.

[126] Y. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar, "Thumos challenge: Action recognition with a large number of classes," *http://crcv.ucf.edu/THUMOS14*, 2014.

[127] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, IEEE, 2014.

[128] R. Girshick, "Fast r-cnn," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1440–1448, 2015.

[129] W. Chen, C. Xiong, R. Xu, and J. Corso, "Actionness ranking with lattice conditional ordinal random fields," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 748–755, 2014.

[130] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[131] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[132] L. Wang, Y. Qiao, X. Tang, and L. Van Gool, "Actionness estimation using hybrid fully convolutional networks," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[133] J. C. SanMiguel and S. Suja, "Liris harl competition participant," 2012. Video Processing and Understanding Lab, Universidad Autonoma of Madrid, Spain, `http://liris.cnrs.fr/harl2012/results.html`.

[134] Y. He, H. Liu, W. Sui, S. Xiang, and C. Pan, "Liris harl competition participant," 2012. Institute of Automation, Chinese Academy of Sciences, Beijing `http://liris.cnrs.fr/harl2012/results.html`.

[135] C. Wolf, J. Mille, E. Lombardi, O. Celiktutan, M. Jiu, E. Dogan, G. Eren, M. Baccouche, E. Dellandrea, C.-E. Bichot, C. Garcia, and B. Sankur, "Evaluation of video activity localizations integrating quality and quantity measurements," *In Computer Vision and Image Understanding*, vol. 127, pp. 14–30, 2014.

[136] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, Ieee, 2009.

[137] P. Mettes, D. Koelma, and C. G. M. Snoek, "The imagenet shuffle: Reorganized pre-training for video event detection," in *Proceedings of the ACM International Conference on Multimedia Retrieval*, (New York, USA), 2016.

[138] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[139] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[140] A. Milan, K. Schindler, and S. Roth, "Multi-target tracking by discrete-continuous energy minimization," *IEEE Transection on Pattern Analysis and Machine Intelligence (PAMI)*, 2016.

[141] B. Majecka, "Statistical models of pedestrian behaviour in the forum," *Master's thesis, School of Informatics, University of Edinburgh*, 2009.

[142] G. Singh, "Categorising the abnormal behaviour from an indoor overhead camera," *Bachelor's thesis, VIT University*, 2010.

[143] R. Šrámek, B. Brejová, and T. Vinař, "On-line viterbi algorithm and its relationship to random walks," *arXiv preprint arXiv:0704.0062*, 2007.

[144] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, pp. 740–755, Springer, 2014.

[145] R. J. Elliott, L. Aggoun, and J. B. Moore, *Hidden Markov models: estimation and control*, vol. 29. Springer Science & Business Media, 2008.

[146] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2017.

[147] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, p. 4, 2017.

[148] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[149] K. Hara, 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*, pp. 18–22, 2018.

[150] C. Xu, C. Xiong, and J. J. Corso, "Streaming hierarchical video segmentation," in *Proceedings of the European Conference Computer Vision (ECCV)*, pp. 626–639, Springer, 2012.

[151] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[152] M. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Proc. of the Eighth Annual Conference of the Cognitive Science Society (Erlbaum, Hillsdale, NJ), 1986*, 1986.

[153] R. Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.

[154] Q. V. Le, N. Jaitly, and G. E. Hinton, "A simple way to initialize recurrent networks of rectified linear units," *arXiv preprint arXiv:1504.00941*, 2015.

[155] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

[156] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato, "Learning longer memory in recurrent neural networks," in *International Conference on Learning Representations*, 2014.

[157] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in *International Conference on Machine Learning*, pp. 1120–1128, 2016.

[158] B. Chang, M. Chen, E. Haber, and E. H. Chi, "AntisymmetricRNN: A dynamical system view on recurrent neural networks," in *International Conference on Learning Representations*, 2019.

[159] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.