Spatio-temporal Human Action Detection and Instance Segmentation in Videos

Suman Saha

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

Department of Computing and Communication Technologies
Artificial Intelligence and Vision Research Group
School of Engineering, Computing and Mathematics
Oxford Brookes University

Supervised by :
Prof. Fabio Cuzzolin, Prof. Nigel Crook and Dr. Tjeerd Olde Scheper

January 2018

Thesis/Assessment Committee

Professor Mubarak Shah (External Examiner)
Director of Center for Research in Computer Vision (CRCV)
University of Central Florida

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

Dr Faye R Mitchell (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, Prof. Nigel Crook and Dr. Tjeerd Olde Scheper between September 2014 and January 2018. 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:

- S. Saha, G. Singh, M. Sapienza, P. H. S. Torr, and F. Cuzzolin, Deep Learning for Detecting Multiple Space-time Action Tubes in Videos, proceedings of the British Machine Vision Conference (BMVC) 2016.
- S. Saha, G. Singh, M. Sapienza, P. H. Torr, and F. Cuzzolin, Spatio-temporal Human Action Localisation and Instance Segmentation in Temporally Untrimmed Videos, arXiv preprint arXiv:1707.07213,2017.
- S. Saha, G. Singh, and F. Cuzzolin, AMTnet: Action-micro-tube Regression by End-to-end Trainable Deep Architecture, proceedings of the IEEE International Conference on Computer Vision (ICCV) 2017.
- G. Singh, S. Saha, and F. Cuzzolin, Online Real-time Multiple Spatio-temporal Action Localisation and Prediction on a Single Platform, proceedings of the IEEE International Conference on Computer Vision (ICCV) 2017.

The author has contributed to the following publications during his PhD, but they are not included in this thesis:

- S. Saha, R. Navarathna, L. Helminger, R. Weber, Unsupervised Deep Representations for Learning Audience Facial Behaviors, Computer Vision and Pattern Recognition (CVPR) workshop, Salt Lake City United States, June 2018.
- F. Cuzzolin, M. Sapienza, P. Esser, S. Saha, M. M. Franssen, J. Collett, H. Dawes, Metric learning for Parkinsonian identification from IMU gait measurements, Journal Gait & Posture 54 (2017): 127-132.

Dedication

In memory of my father
Late *Shri Gour Chandra Saha*
With love and eternal appreciation

I would like to dedicate this work to my parents, sisters, wife Sunita (Tika) and my son Aman. Without such loving and caring people around me, it would not have been possible for me to learn both Science and general aspects of life in the course of these three and a half years journey.

"I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."

[Alan Turing, Computing machinery and intelligence]

Acknowledgements

First and foremost I would like to express my heartfelt thanks to my supervisors Prof. Fabio Cuzzolin, Prof. Nigel Crook and Dr. Tjeerd Olde Scheper for providing such an exceptional opportunity for me to pursue a doctorate in 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 inspire me to push myself beyond my limits and allow 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 has given me inspiration to work in the Artificial Intelligence and Vision Group, 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 those long technical discussions at TVG lab. Also I am grateful to Dr. Sapienza for his consistent guidance, help and support throughout the first two years of my PhD. A special thanks to my lab mate and co-author Gurkirt Singh for his close collaboration and daily discussions led to several exciting works and three publications in BMVC 2016 and ICCV 2017 (which are considered as the top ranked conferences in Computer Vision).

I want to thank: Cigdem Sengul and Tjeerd Olde Scheper for their invaluable guidance and support as research tutors; Jill Organ, Catherine Joyejob, Lorna Denby, Lynn Farrell, Amanda Holden who helped me in all administrative tasks; Gerald Roy for excellent technical, software and hardware support. I would like to thank the MSc students I guided during my PhD, Misbah Munir, Kurt Degiorgio, Stavros Gasparis, Manuele Di Maio, Shashwat Shukla (B. Tech. IIT Bombay, India). It was a great and valuable experience. I am grateful to all my fellow colleagues and friends I met over the past few years: Ruomei Yan, Jalawi Alshuduki, Cristian Roman, Luis Fuente Fernandez, Paul Sturgess, Basel Yousef, Bedour Alshaigy, Mireya Munoz Balbontin, Mohamed Idries, Alla Vovk, Will Guest, at the TDE department, School of Engineering, Computing and Mathematics, Oxford Brookes University.

My special thanks go to Dr. Romann Weber for offering me an internship at Disney Research Zurich (DRZ), Switzerland - *it was really exciting!* Many thanks to Romann for his invaluable time and guidance during those long hours

technical discussion at DRZ lab. I feel really fortunate to get a chance to work in his Machine Learning group. Special thanks go to my DRZ lab mates for their help and support: Rajitha Navarantha, Leonhard Helminger, Johannes Brand, Leon Palaic, Ivan Ovinnikov, Gerhard Rothlin. The experience I gained at DRZ is extremely valuable, and help me to understand and apply some very recent cutting-edge Deep Learning algorithms in Computer Vision domain.

Due acknowledgment goes to my MSc supervisors Dr. Ashutosh Natraj and Dr. Sonia Waharte for their time, advice and support during my MSc dissertation, and for offering me to work as a research intern at the Department of Computer Science, University of Oxford. It is unthinkable for me to write acknowledgement without mentioning a few names of my colleagues I worked with for past few years at the Research and Development and Scientific Services division, Tata Steel Limited, Jamshedpur, India. I am thankful to my Tata Steel colleagues for inspiring me to pursue a PhD: Dr Sumitesh Das, Dr Abhishek Raj, Dr. Raju Venkat Dasu, Dr. Shoaib Jameel. My heartfelt thanks go to Sumitesh Das (Chief of R&D Tata Steel Ltd. India) and Nagaraja B.N. (HOD Computer Science Dept. Siddaganga Polytechnic, India) for providing their references to support my PhD application at Oxford Brookes University, UK.

Last but not the least, I am thankful to my *sisters* and *parents* in India for their generosity; my friend *Debleena Chaudhuri* for her consistent moral support, and my wife *Sunita* for her unbounded patience!

Abstract

With an exponential growth in the number of video capturing devices and digital video content, automatic video understanding is now at the forefront of computer vision research. This thesis presents a series of models for automatic human action detection in videos and also addresses the space-time action instance segmentation problem. Both action detection and instance segmentation play vital roles in video understanding.

Firstly, we propose a novel human action detection approach based on a frame-level deep feature representation combined with a two-pass dynamic programming approach. The method obtains a frame-level action representation by leveraging recent advances in deep learning based action recognition and object detection methods. To combine the the complementary appearance and motion cues, we introduce a new fusion technique which significantly improves the detection performance. Further, we cast the temporal action detection as two energy optimisation problems which are solved using Viterbi algorithm.

Exploiting a video-level representation further allows the network to learn the inter-frame temporal correspondence between action regions and it is bound to be a more optimal solution to the action detection problem than a frame-level representation. Secondly, we propose a novel deep network architecture which learns a video-level action representation by classifying and regressing 3D region proposals spanning two successive video frames. The proposed model is end-to-end trainable and can be jointly optimised for both proposal generation and action detection objectives in a single training step. We name our new network as *"AMTnet"* (**A**ction **M**icro-**T**ube regression **N**etwork). We further extend the AMTnet model by incorporating optical flow features to encode motion patterns of actions.

Finally, we address the problem of action instance segmentation in which multiple concurrent actions of the same class may be segmented out of an image sequence. By taking advantage of recent work on action foreground-background segmentation, we are able to associate each action tube with class-specific segmentations.

We demonstrate the performance of our proposed models on challenging action detection benchmarks achieving new state-of-the-art results across the board and significantly increasing detection speed at test time.

Extended Abstract

With an exponential growth in the number of video capturing devices and digital video content, automatic video understanding is now at the forefront of computer vision research. This thesis presents a series of models for automatic human action detection in videos. One of the models presented in this work also addresses action instance segmentation alongside detection. Both action detection and instance segmentation play vital roles in video understanding. Systems capable of detecting human actions have far-reaching applications such as Google-style video indexing and retrieval, autonomous driving, robot-assisted surgery, human-robot interaction, virtual reality gaming to name but a few.

Human action detection consists in classifying actions present in a video, such as "clapping", "running", and simultaneously predicting the spatial and temporal extents of these actions. The existing promising deep feature based approaches perform action detection at the cost of using expensive multi-stage detection frameworks and unsupervised region proposal algorithms. Moreover, either temporal detection is not addressed at all, or solved using an expensive sliding-window scheme. To overcome these issues, we propose a novel approach based on a frame-level deep feature representation combined with a two-pass dynamic programming approach. The method obtains a frame-level action representation by leveraging the two-steam CNN architecture where an appearance (RGB) and a motion (optical flow) stream are trained separately for frame-level action classification and 2D proposal regression. To combine the complementary appearance and motion cues, we introduce a new fusion technique which significantly improves the detection performance. Further, we cast the temporal action detection as two energy optimisation problems which are solved using Viterbi algorithm. Unlike previous approaches, our model uses an inexpensive single-stage pipeline, supervised region proposals and replaces sliding window with dynamic programming. We demonstrate the performance of our approach on challenging action detection benchmarks achieving new state-of-the-art results across the board and significantly increasing detection speed at test time.

Exploiting a video-level representation further allows a network to learn those inter-frame temporal associations (between action regions) which can not be learnt using a frame-level representation, and thus, with frame-level features we have to entirely rely on a post processing step to connect per frame detections in time. To this end, we propose a novel deep network architecture which learns a video-level action representation by classifying and regressing 3D region pro-

posals spanning two successive video frames. The proposed model is end-to-end trainable and can be jointly optimised for both proposal generation and action detection objectives in a single training step. At test time the network predicts "micro-tubes" encompassing two successive frames, which are linked up into complete action tubes via a new algorithm which exploits the temporal encoding learned by the network and cuts computation time by 50% (Chapter 7). We name our new network as *"AMTnet"* (**A**ction **M**icro-**T**ube regression **N**etwork). Promising results on standard benchmarks show that AMTnet does outperform the state-of-the-art when relying purely on appearance.

We further extend the AMTnet architecture by incorporating a motion stream and introduce a new fusion technique which fuses RGB and flow CNN features at training time, allowing the network to learn pixel-wise correspondences between appearance and motion features. We name this new deep network as *"AMTnet-Flow"*. We quantitatively demonstrate that the AMTnet-Flow outperforms the top competitor in video-mAP by large margins of 6.28% and 4.79% on higher IoU thresholds 0.5 and [0.5:0.95] respectively on one of the most challenging action detection benchmarks. Moreover, we introduce a new bounding-box interpolation algorithm for extracting longer detection micro-tubes which significantly improves the detection speed at test.

Unlike action detection, an instance segmentation method can accurately localise actions at a finer pixel-level by assigning a class- and instance-aware label to each pixel. We address the problem of action instance segmentation in which multiple concurrent actions of the same class may be segmented out of an image sequence. By taking advantage of recent work on action foreground-background segmentation, we are able to associate each action tube with class-specific segmentations. We demonstrate the performance of our algorithm on the challenging benchmark and achieve a new state-of-the-art result which is 14.3 times better than previous methods.

Contents

List of Figures

List of Tables

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

The spatio-temporal human *action detection* [1] problem in computer vision refers to the following three sub-problems: 1) action classification, 2) spatial detection and 3) temporal detection. Given an input video, the detection model is supposed to classify each action instance, localise them spatially using bounding boxes and detect the temporal extent of each action instance by inferring its start and end time points.

Consider Figure 1.1 (**top row**) an input video sequence to the detection system, i.e. frames at different time steps from a *"biking"* video sequence. The number on the top of each video frame denotes the frame number. Figure 1.1 (**bottom row**): the output of an ideal detection system, i.e. the predicted class labels, bounding boxes (spatial detection) and temporal extents (temporal detection) of three different *"biking"* action instance. Each colour represents an action instance.

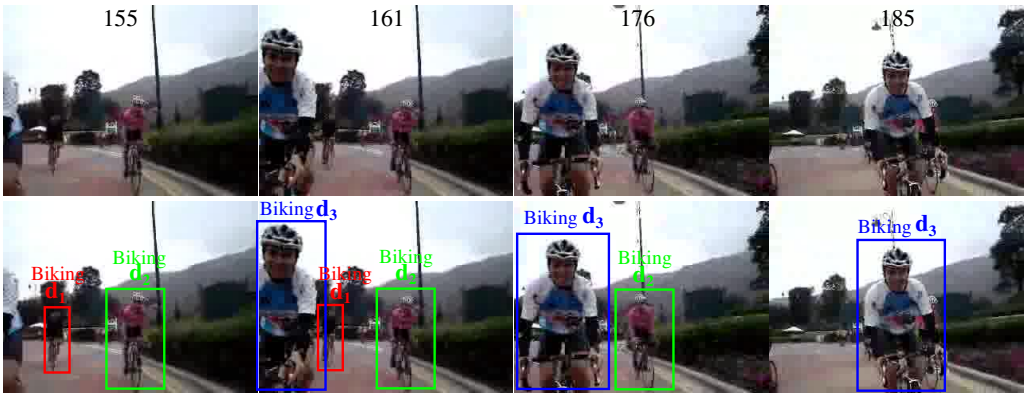


Figure 1.1: ***Top row****: input video sequence* ***Bottom row****: output of an ideal detection system.*

[1] Action detection is also called action localisation in the literature.

Action instance segmentation, in contrast, is the problem of detecting and delineating each distinct action instance present in a video [1]. In other words, we assign a label to each pixel of the frame where labels are class-aware and instance-aware. Consider again the same input video sequence as shown in Figure 1.1 **top row**). The output of an ideal detection system capable of performing both action detection and instance segmentation is depicted in Figure 1.2.

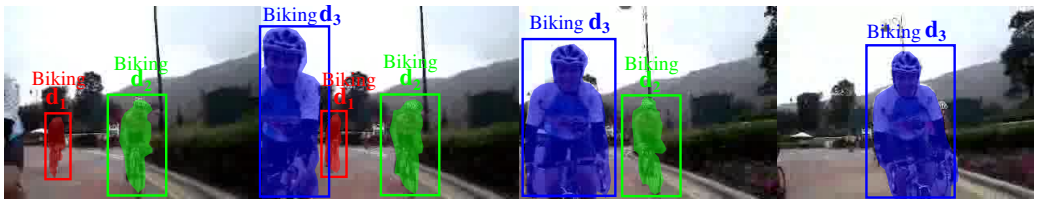


Figure 1.2: Output of an ideal detection system capable of performing action instance segmentation. Note that, the system is able to uniquely identify the three different instances of a "Biking" action class and assign each pixel (belongs to an action instance) a class- and instance-aware label where the colours red, green and blue represent the three unique instance-aware labels.

1.1 Motivating Real world applications of action detection

With an exponential growth in the number of video capturing devices (CCTV cameras, smartphones), uploaded videos on the web (e.g. YouTube, FaceBook, DailyMotion) [2] and Internet users [3] [2–4], automatic video understanding has become the forefront of computer vision research. Thus, there is an increasing demand for a tool which can automatically understand (or analyse) this massive amount of video data.

Automatic detection of human actions will allow a Google-style video search and retrieval. Figure 1.3 illustrates an example of real-world scenario where an action detection system is useful. A user wants to retrieve video subsets from an online repository of huge video data. In particular, he wants to access only those video frames where action "basketball dunk" happens. Such a video retrieval system requires action detection capabilities to recognise the "basketball dunk" action and localise its spatial and temporal extents within a video. Another interesting application could be a movie recommendation system where automatic

[2] https://www.youtube.com/yt/press/statistics.html
[3] http://www.reelseo.com/

action detection of audiences within a movie theater may help in better understanding the audience group behaviour resulting high quality recommendation system beneficial to writers, directors, marketers, advertisers [5, 6]. Significant

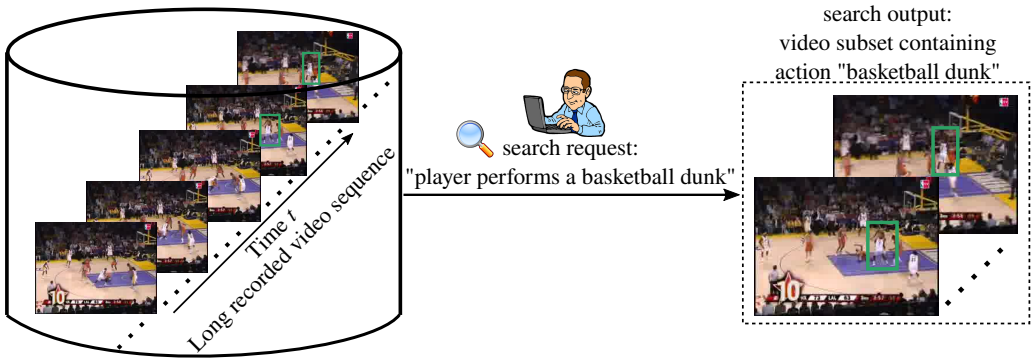


Figure 1.3: Based on the search "basketball dunk" the system detects video frames where this action happens.

progress has been made in image understanding e.g. object detection and segmentation, image captioning, face recognition. However, for video undersanding, we need effective tools to analyse raw video data, and in particular human actions.

The action detection algorithms can be categorised as either *"offline"* or *"online"*. Examples of motivating real-world applications of an automatic human action detection system is shown in Figure 1.4. Those detection systems which require the entire video clip beforehand to generate detection results fall into the offline category, and those which are able to generate detections from a small subset of the video and can incrementally link those detections over time as more and more frames are visible to the system are categorised as online. Offline algorithms are suitable for applications such as video indexing and retrieval. On the other hand, applications such as self-driving cars, robot-assisted surgery and human-robot interaction require online algorithms to process the streaming video frames as they arrive. In autonomous driving [7,8], the space-time localisation of human actions will allow the system to detect different categories of actions, e.g. *"walking"* and *"running"*, thus enabling the vehicle controller to adjust speed and course accordingly. In robot-assisted surgery [4] [9–11], an accurate action detector enables the robot to provide the surgeon an environment to carry out complex procedures with better precision and control. The quality of human-robot interaction and virtual reality gaming can be improved with smarter action

[4]Our group head, AI and Vision research group, was awarded the Horizon 2020 project, on the development of robotic assistant surgeons for laparoscopy: `http://cms.brookes.ac.uk/staff/FabioCuzzolin/`

detection algorithms.

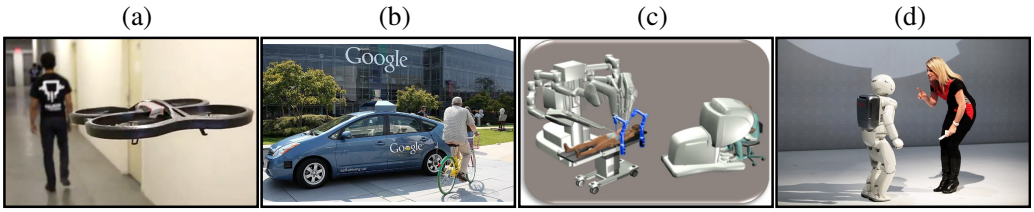


Figure 1.4: Examples of motivating real-world applications of an automatic human action detection system. ***(a)*** *autonomous drones,* ***(b)*** *self-driving cars,* ***(c)*** *robot-assisted surgery,* ***(d)*** *human-robot interactions.*

Furthermore, instance segmentation can significantly enhance the quality of the action detection by providing (a) more accurate pixel level (as opposed to bounding-box level) localisation results and (b) class- and instance-aware (as opposed to only class-ware) labels to delineate each instance of the same action. The greater part of this dissertation is focused on action detection, whilst in Chapter 6 we introduced an detection pipeline which can perform both action instance segmentation and detection.

The action detection pipelines presented in Chapter 5 and Chapter 6 are suitable for offline applications, as they follow a multi-stage detection process and are relatively expensive. In contrast, Chapter 7 and Chapter 8 are devoted to designing faster and single-stage end-to-end trainable action detection frameworks which can be easily extended to online versions just by replacing their offline action tube generation algorithms with the online tube building algorithm [12].

1.2 What is a spatio-temporal action instance?

Before answering the above question, we might be interested to know even: *"what is an action?"*. A very nice explanation of this question can be found in Section 1.2 of [13]. An action can be defined in two different perspectives, depending on whether the definition is tailored for a computer or a human. Here we are interested in defining an action for computers. From a computer's perspective, in order to explicitly programme for an action, it must have a precise definition. However, due to high complexity and variation associated with human actions, it is impractical to define an action down to small details, and difficult to come up with a well thought out definition which can be programmed by a computer. Due to these aforementioned reasons, researchers often use their own definition tailored for their application. In this way, we can say that an action is simply what you defined it to be. In this thesis, we adopt the definition of human actions

provided by the standard benchmarks in action detection (Chapter 4). Some examples of human actions defined by these benchmarks are: *"run"*, *"jump"*, *"biking"*, *'horse ridding"*, *"handshake"* etc.

In a given input video, an action may be present in any spatial and temporal locations. Besides, multiple co-occurring instances of the same or different action class can be present in a video. Consider Figure 1.5 (a) and (b) in which multiple action instances of the same class *"fencing"* (4 instances) and *"biking"* (3 instances) appear in the respective video frames. In Figure 1.5 (c) and (d), multiple action instances of the two different classes *"handshaking"*, *"leave baggage unattended"* and *"unlock enter/leave room"*, *"put/take object into/from box/desk"* are present respectively. As anticipated, bounding boxes are used to spatially localise each action instance in a video frame. Each colour represents a unique action instance, no matter the class it belongs to. The set of bounding boxes connected over time without any whole/gap in between forms a spatio-temporal action instance as shown in Figure 1.5 (e). In the action detection literature, each spatio-temporal action instance is typically termed an *"action tube"* [14]. Note that each action tube also tells us about the temporal duration/extent of that action, i.e. the start and end times.

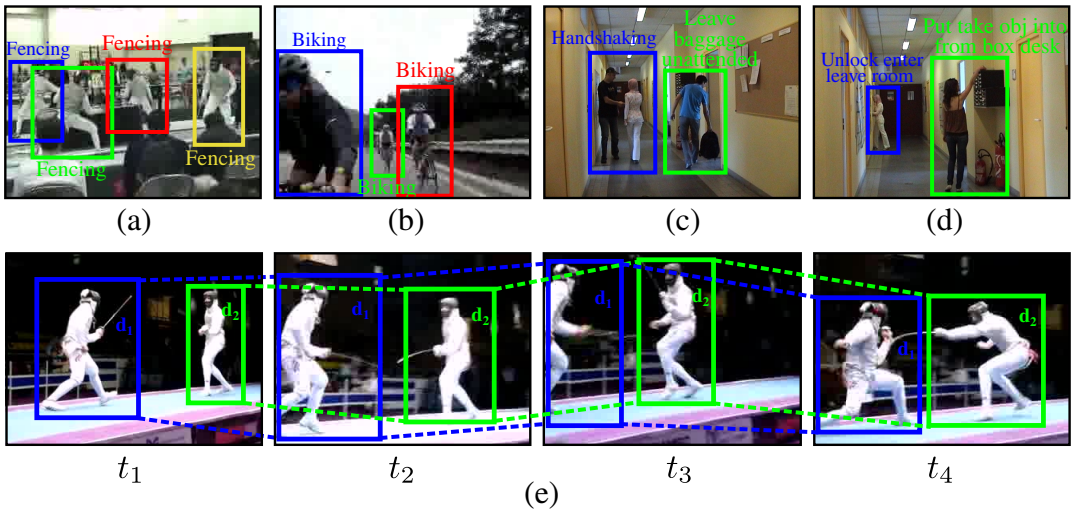


Figure 1.5: Spatial localisation of action instances of class: ***(a)*** *"fencing" (4 instances),* ***(b)*** *"biking" (3 instances),* ***(c)*** *"handshaking" and "leave baggage unattended",* ***(d)*** *"unlock enter leave room" and "put take object into from box desk".* ***(e)*** *spatio-temporal localisation of two "Fencing" action instances, where t denotes time and $t_1 < t_2 < t_3 < t_4$. Dotted lines represent temporal association between frame-level detections over time.*

1.3 Challenges in spatio-temporal human action detection

Spatio-temporal human action detection is a hard problem. It is extremely challenging to design a representation (for a machine to understand) which is robust enough to deal with both *intra-class variability* and *inter-class confusion* problems. Other major difficulties arise from spatial and temporal localisation, and Inter-frame data association. In addition, human actions possess high variations in geometry and topology [15].

1.3.1 Intra-class variability

The appearance and motion of a particular class may differ significantly due to variation in illumination, camera motion and viewing angle, partial occlusions, background, scale. In addition, action instances belonging to the same class may differ due to a large variability in the execution style of an action in terms of poses, motion dynamics, contextual information, speed etc. For instance, the two action instances in Figure 1.6 (a) and (b) exhibit variation in pose, camera viewing angle, contextual information (a door and a bench) yet they belong to the same action class "Sit".

1.3.2 Inter-class confusion

There are cases in which instances from different action categories possess similar: (a) appearances, (b) contextual information (e.g. players in both *"basketball"* and *"basketball dunk"* look the same, and in both actions context is provided by the presence of basketballs, (c) motions and (d) poses (e.g. actions *"stand"* and *"sit"*). Some other interesting examples are: run and walk, laugh and yawn, crawl and swim where it is hard to generate a discriminative representation.

Thus, learning a representation which has good generalisation ability over a wide range of action instances belonging to the same class, and yet able to discriminate between actions of different classes is challenging [16]. Figure 1.6 highlights some of the above problems (video frames are taken from action detection datasets: UCF-101-24 [17] and J-HMDB-21 [18]).

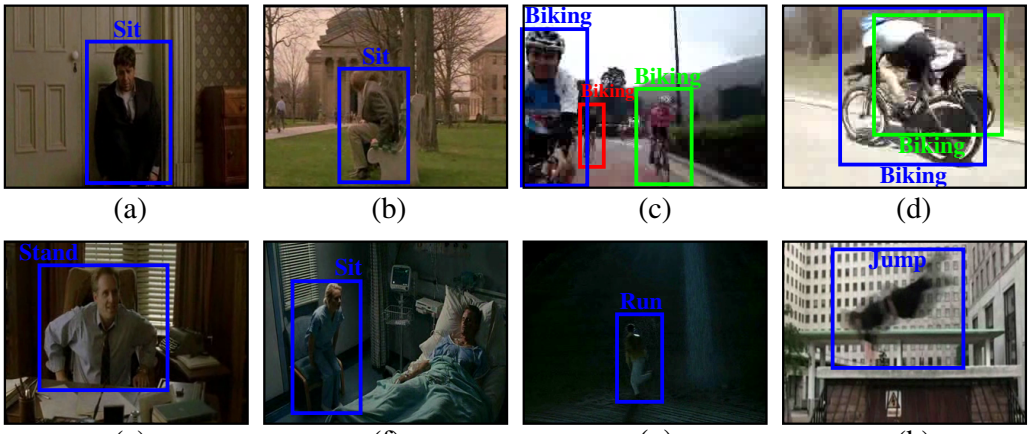


Figure 1.6: ***(a-b)****: intra-class variability - the two action instances show variation in pose, camera viewing angle, contextual information (a door and a bench) yet they belong to the same action class "Sit".* ***(c-d)****: in (c), the "Biking" instance depicted with a red bounding box is hard to detect due large variation in scale; in (d), the "Biking" instance localised by a green coloured bounding box is partially occluded by the blue coloured one and thus difficult to detect.* ***(e-f)****: inter-class confusion - although these two action instances share similar poses, motions, contextual information (chairs) they belong to two different classes "Stand" and "Sit".* ***(g-h)****: these action instances are hard to detect due to poor illumination (g) and motion blur (h), respectively.*

1.3.3 Spatial localisation

Finding the spatial locations of action instances present in a video is challenging due to several factors: (a) they may vary over time, (e.g. the locations of the "biking" instance d_3 and the "fencing" instance d_1 change over time as shown in Figure 1.1 and Figure 1.5 respectively; (b) the actor is partially occluded by other actor or object (e.g. the one of the bikers (located with green bounding box) is partially visible due to occlusion as shown in Figure 1.6 (d)); (c) co-occurring action instances belonging to the same or different action classes may be present (Figure 1.5); (d) in addition, an actor can perform multiple actions at the same time (e.g. a person is "waving his hand" while "walking").

1.3.4 Inter-frame data association

Consider Figure 1.7. It shows a video with a sequence length of 80 frames (f_1 to f_{80}) in which there are two action instances. To correctly detect these two instances, one approach would be to associate or link the frame-level detections (d_1 and d_2) in time to obtain two valid action tubes. The temporal linking of detections in action detection can be assimilated to the data association problem [19] in the *"multi-target tracking"* domain [20]. However, in tracking data

association is done for detections of a single class such as *"pedestrian"*, whereas in action detection data association is to be performed for detections belonging to multiple human action categories. In Figure 1.7, the correct inter-frame data associations for each action instance are depicted using blue and red solid lines. Data association is hard because of the following main reasons. (a) As the number of frame-level detections m and the sequence length n of an input video increase the running time complexity of the data association algorithm also increases - e.g. in the case of a Viterbi algorithm the complexity increases in polynomial time, the running time is $\mathcal{O}(nm^2)$ [21]. (b) Another challenge is to make the data association algorithm robust to position-swapping (Figure 1.7) and associate the inter-frame detections in time correctly (in the figure, the locations of action instances d_1 and d_2 are interchanged between frame f_{20} to f_{40}). (c) Lastly, the data association problem is made harder by partial occlusion, sudden-change in illumination, camera motion and/or viewing angle.

1.3.5 Temporal localisation

Modelling a system which can correctly infer the temporal extent (start and end time points) (Figure 1.7) of each action instance present in a video is extremely difficult - action instances may appear and disappear within a video at any time point. Consider Figure 1.1. The *"biking"* action instances d_1 and d_2 disappear after frames 161 and 176 respectively. Whereas, d_3 starts at frame 161. Even though the actor is present throughout the entire video, the actual start and end time of the action may be at any time point within the duration of the video (Figure 1.8) which make the temporal detection problem harder. A naive sliding window algorithm for detecting the temporal extent of actions will have a running time $\mathcal{O}(n^2)$ where n is the number of frames in a video.

1.4 Dissertation outline

In spite of these aforesaid challenges in action detection, substantial research initiatives have been taken in this area [14, 20, 22–26]. For a detailed list of works, please refer Chapter 3).

First and foremost, we present the avenues of investigation in Chapter 2 which provides insights into the evolution of the action detection problem. Next, we introduce the human action detection datasets used throughout this work in Chapter 4. We then formulate a spatio-temporal action detection approach in

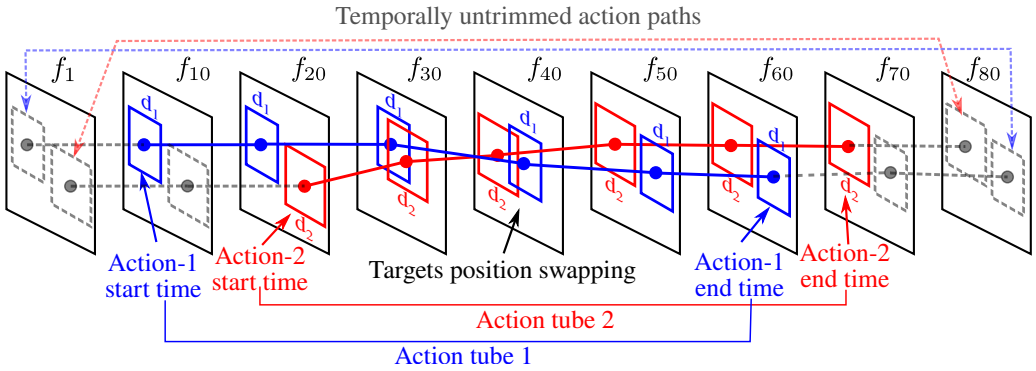


Figure 1.7: Linking frame level bounding boxes to build two action tubes. Each action tube is assigned a unique id, here ids are 1 and 2. Two different colours (red and blue) are used to denote two different action tubes. Solid lines show valid detections and temporal links whereas, the dotted lines denote detections and links discarded during temporal localisation of action instances. Note that, in frame 40 (f_{40}) the tube ids get swapped which is hard to track. Also note that the start and end times are different for two action tubes, and detecting these temporal extent/duration of each tube is again challenging.

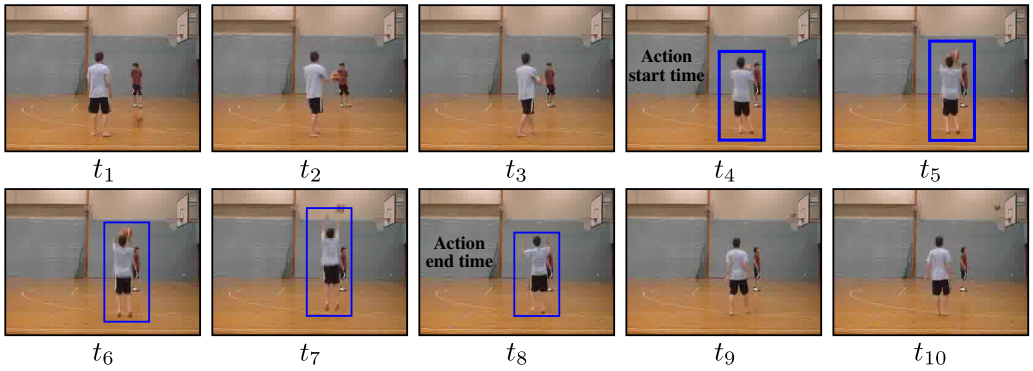


Figure 1.8: Temporal detection problem: Even though the actor is present throughout the entire video the actual "basketball" action starts at t_4 and ends at t_8 which makes the temporal detection more difficult.

Chapter 5 in which frame-level RGB and optical flow image data are mapped to a feature space using a two-stream convolutional neural network (CNN) architecture [27] and subsequently, these CNN features are used for action classification and bounding box regression. In addition to predicting the frame-level confidence score and spatial location of each action instance in a video, the proposed framework also links the detections in time and apply temporal label smoothing to solve for temporal localisation.

In Chapter 6 we consider the problem of spatio-temporal action instance segmentation, where the task is to detect and delineate each action instance by assigning each pixel a label. Unlike the bounding-box level class-aware action labels in Chapter 5, the labels obtained from instance segmentation are at pixel-

level and they are both class- and instance-aware.

We present a more appropriate solution to the action detection problem in Chapter 7. Note that the framework presented in Chapter 5 can only provide a suboptimal solution to the problem as it relies on training CNNs which can predict only frame-level class confidence scores and bounding boxes. As these CNNs are trained on individual video frames, they do not learn the temporal associations between inter-frame action detections. The new action detection framework presented in Chapter 7 addresses this limitation by training CNNs on pairs of video frames and predicting video-level confidence scores and detections (i.e. action "micro-tubes" (Section 2.3)).

An extension of the work in Chapter 7 to incorporate motion features, and predict action micro-tubes over more widely separated pairs of videos frames is presented in Chapter 8. The detection frameworks presented in Chapter 7 and Chapter 8 are computationally efficient, faster and both adopt single-stage end-to-end training and testing strategies. In contrast, the detection pipelines presented in Chapter 5 and Chapter 6 are relatively expensive and slower, follow multi-stage training and testing strategies and are thus not suitable for online applications. *As we move along the chapters the state of the art was changing in time, and as a result we are comparing ourselves with a moving bar.*

1.5 Contributions of this thesis

Firstly, the main contributions of the work presented in Chapter 5 are: (1) a novel action tube formulation, (2) a new deep learning framework for powerful action representation, and (3) an efficient fusion scheme for fusing appearance and motion cues. Unlike the previous state-of-the-art approach [20] which solves the temporal action localisation using an expensive sliding window scheme, our action tube formulation uses an efficient two-pass Viterbi algorithm for temporal detection. Moreover, by leveraging the latest advancements in object detection, our deep network learns better action representation than the previous work [14, 20]. The detailed contributions of this chapter are as follows. I cast the expensive multi-stage action detection approach [14, 20] (Section 2.1) into a relatively less expensive single-stage setting where appearance- and motion-based CNNs are trained on RGB and optical flow data to perform frame-level action classification and spatial localisation. Subsequently, frame-level RGB and flow detections are fused and then linked in time to generate class-specific action paths. Finally, label smoothing is applied to each action path to generate In or-

der to achieve a better representation of image data and to design a single-stage detection framework, I replace the R-CNN architecture [28] used in [14, 20] with Faster R-CNN [29]. The relatively deeper representation of VGG-16 [30] (used in Faster R-CNN) is beneficial for the classification accuracy and allows our model to generalise well to a wide range of action detection datasets. I then report the outperforming action classification results (Section 5.4.2) of our framework on the J-HMDB-21 action detection dataset [18]. As a by-product of using Faster R-CNN, our detection network has access to relatively better quality action proposals (predicted by a RPN network) than those generated by the Selective Search (SS) algorithm [14]. I present a quantitative comparision (Section 5.4.4) between SS- and RPN-based region proposals and demonstrate that RPN-based proposals exhibit much better recall-to-IoU than SS-based boxes. Furthermore, I propose an original test time fusion strategy to fuse the RGB and optical flow information based on the detections' softmax probability scores and their inter-frame spatial overlaps (Section 5.3.3). I demonstrate quantitatively that our fusion strategy significantly improves the detection accuracy (Section 5.4). Besides, I report an ablation study (Section 5.4.5) which again support the significance of our fusion method. Further, (a) to link the frame-level detections (after fusion) over time and (b) to perform temporal localisation, I bring forward a two-pass dynamic programming (2PDP) approach, in which the Viterbi algorithm is used to solve optimisation problems (a & b). I also demonstrate the efficacy of our temporal detection algorithm (the 2^{nd} pass of DP) which significantly improves the action detection performance (Section 5.4.6).

Secondly, in Chapter 6 I propose a variant of the above action detection pipeline which can perform action instance segmentation alongside action detection. To allow our model to output pixel-level instance segmentation, I apply the human motion segmentation algorithm [31] on the testset videos to extract binary silhouettes of human actions in space and time. I then propose a simple but effective region proposal algorithm (Section 6.3.1) which generates region proposals based on the power set of connected components in the space-time binary silhouettes of human actions. To the best of my knowledge, *this is the first work which addresses jointly both space-time action instance segmentation and action detection problems*. I also demonstrate quantitatively that the proposed model outperforms the existing methods in one of the most challenging action detection datasets available to date (Section 6.4). Moreover, I am the first to show qualitative action instance segmentation results.

As a third significant contribution, I propose a new action detection paradigm

in Chapter 7 which represents a key conceptual step forward from the frame-level action representation (Section 2.1) (heavily exploited in [14, 20, 32, 33]) towards a video-level representation (Section 2.2). In place of encoding frame-level action regions to a feature space, I encode video-level action regions (i.e. action regions span across a pair of successive frames) to a feature space using a fusion technique (Section 7.3.1) which performs element-wise fusion of convolutional features computed from two successive video frames. Unlike the frame-level representation, the video-level feature encoding can effectively encodes the temporal associations between action instances present in a video subset. On the network design side, I propose a novel end-to-end trainable deep neural network architecture which facilitates video-level feature encoding. This new network architecture also addresses region proposal generation and action detection task jointly using a single round of optimisation (Section 7.4). One of the core building blocks of this new architecture is a 3D regional proposal network (RPN) (Section 7.3.3). I design a 3D-RPN which generates space-time video region hypotheses in place of frame-level 2D proposals. Unlike what happens in standard action detection approaches [14, 20, 32, 33], I implement a simple but efficient regression technique for regressing such 3D proposals (Section 7.4.1). The output of this new action detection network is a set of action micro-tubes (Section 2.3) instead of a set of 2D detection windows as in [14, 20, 32, 33]. I also propose a new action tube generation algorithm suitable for connecting these micro-tubes so generated, which exploits the temporal encoding learnt by the network (Section 7.5). I demonstrate quantitatively that our model outperforms state-of-the-art appearance-based models, while being highly competitive with methods which exploit both appearance and motion features (Section 7.6). Moreover, to the best of my knowledge, in the action detection community *I am the first to apply "bilinear interpolation"* [34, 35] (instead of a max-pooling [29]) for ROI [5] feature pooling (Section 7.3.4). A bilinear interpolation layer allows the gradients of the loss to flow backwards with respect to both the inputs (a) convolutional features and (b) coordinates of bounding boxes. Thus, using bilinear interpolation layer, affine or morphed region proposals can be predicted in place of rectangular windows [36]. We keep this extension as a future work.

Finally, in Chapter 8 an extension of Chapter 7 where the action detection performance and speed are significantly improved by leveraging optical flow based deep features and a training strategy which incorporates both long and

[5] A ROI (region of interest) is a rectangular bounding box parameterized as 4 coordinates in a 2D plane [$x1$ $y1$ $x2$ $y2$].

short distance frames. More specifically, I propose to add an additional optical flow based motion stream to the existing deep architecture in Chapter 7. I demonstrate that by training the network on both long and short distance video frame pairs, and subsequently testing it on long distance pairs can improve the detection performance and speed (Section 8.5). To deal with micro-tubes generated from long distance frame pairs at test-time, I implement a simple but elegant bounding box interpolation algorithm (Section 8.5.2) which makes the tube generation process relatively faster (Section 8.5.3).

The above contributions and these conceptual steps will lead, in the medium term, to a deep network architecture able to regress whole action tubes which is considered as an optimal solution to the action detection problem.

1.6 Software packages and media

1.6.1 List of Software Packages

The following software packages from this thesis are available online.

- Source code for our BMVC 2016 [33] 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).

- Matlab source code for our action instance segmentation work [37] is available online at:
  `https://bitbucket.org/sahasuman/matvis/` (private access).

- Lua and Torch based source code for our AMTnet work [38] is available online at:
  `https://bitbucket.org/sahasuman/amtnet_iccv2017` (private access).

1.6.2 In the media

The following YouTube videos showcasing various qualitative results of this thesis.

- BMVC 2016 work [33] - YouTube demo video link
  `https://youtu.be/vBZsTgjhWaQ`.

- Action instance segmentation work [37]:
  (a) YouTube demo video link - main paper

https://youtu.be/fqqgFQzmkfM
(b) YouTube demo video link - generation of optical flow trajectories [39]
https://youtu.be/iaZ2x1LqFxA
(c) YouTube demo video link - generation of supervoxels [40]
https://youtu.be/skzG4uolcyw
(d) YouTube demo video link - human action segmentation [41]
https://youtu.be/cPjbjAPm2jo.

Chapter 2

Avenues of investigation

"Features matter"! Powerful visual features or image representation techniques are the key to the success of any computer vision algorithm such as image classification and object recognition systems [42]. Over the past several years, the improvements in the performance of computer vision based systems can be attributed to the evolution of effective data representation starting from early Bag-of-Visual-Words (BoVW) [43, 44], to Improved Fisher Vector (IFV) [45], and more recently, the deep feature representation provided by Convolutional Neural Networks (CNNs) [46]. In recent years, deep feature representations have demonstrated significant quantitative improvements in image understanding over classical ones (e.g. BoVW, IFV etc.) [42].

Unlike still images, videos contain highly dynamic appearance and motion patterns. Therefore, problems such as human action detection in videos require effective visual representation which is robust to the time varying visual data. Further, as discussed earlier (Section 1.3.4), the two main challenges in action detection are: (1) inter-frame data association and (2) temporal action localisation. Most action detection approaches try to solve these problems using a graph-based video segmentation and sliding window technique (Chapter 3) which are computationally expensive and inefficient to deal with real-world longer action sequences.

In this work, we mainly focus on exploiting deep features for image and video data representation and pose inter-frame data association and temporal localisation as energy optimisation problems which can be efficiently solved using the Viterbi algorithm. In the following sections, we introduce the frame- (Section 2.1) and video-level (Section 2.2 & 2.3) deep feature representation used in this work, combined with the original methods proposed (Section 2.4 & 2.5) to address the spatio-temporal action detection problems set out in Section 1.4.

2.1 Deep feature representation of spatial action instances

Unlike action recognition, our goal is to both localise (in space and time) and classify action instances in temporally untrimmed videos. Two promising efforts in this direction were proposed by Gkioxari and Malik [14] and Weinzaepfel *et al.* [20]. They used CNN-based deep feature representation to encode both spatial and temporal features of human actions to detect action tubes. The success of these methods [14, 20] are mainly due to: (1) the use of regions with CNN (R-CNN) features [28] for video data representation which not only provides better feature encoding of human actions but also improves the spatial localisation accuracy with the help of region proposals [47]; (2) use of a two-stream CNN approach [27] to effectively capture the complementary information from video data i.e. frame-level appearance and inter-frame motion dynamics associated with human actions.

However, [14, 20] have a number of drawbacks: 1) they are computationally expensive, 2) require multi-stage training and 3) rely on unsupervised region proposal algorithms such as Selective Search [14] or EdgeBoxes [20]. The main cause of such limitations is that, these action detectors [14, 20] rely on R-CNN object detection framework. Unlike [48, 49], R-CNNs do not share computation and perform a separate forward pass through convolutional layers for each object proposal (see Figure 2.1 (a)), which is expensive in terms of both time and computing resources. For example, during training R-CNNs require to execute 2000 conv forwards passes for 2000 Selective Search proposals, amounting to relatively longer training time and more GPU resources than [48, 49]. Besides, the R-CNN framework follows a multi-stage training pipeline which includes (a) fine-tuning a CNN, (b) extracting CNN features, (c) caching features to disk, (d) training a number of one-vs-all SVMs and finally (e) solving a regression problem to fit the region proposal bounding-boxes as per the ground truths, resulting in a very computationally expensive pipeline. Furthermore, the detection accuracy of R-CNNs is limited by their relying on unsupervised region proposal algorithms [14, 20] which, besides being resource-demanding, cannot be trained for a specific detection task and are disconnected from the overall classification objective.

For instance, on large datasets such as UCF-101 [17], Gkioxari and Malik [14]'s action detection pipeline takes a week for training and feature extrac-

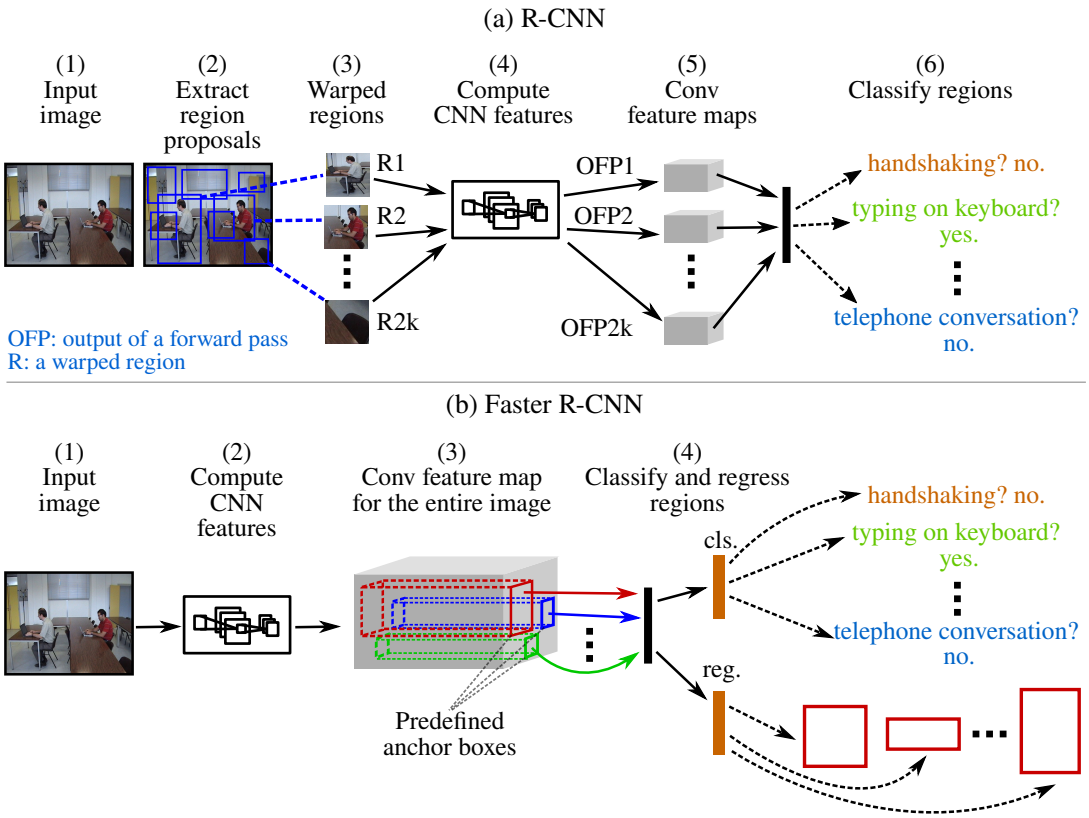


Figure 2.1: ***(a)*** *R-CNN and* ***(b)*** *Faster R-CNN architectural difference.*

tion [1] plus one extra day for SVM training. At test time, detection is slow as features need to be extracted for each region proposal via a CNN forward pass. Moreover, Gkioxari and Malik [14]'s work does not address temporal localisation, whereas Weinzaepfel *et al.* [20]'s sliding window approach for temporal action detection is relatively expensive.

To deal with these limitations, we explored the possibility of modelling a deep feature-based action detection framework which is capable of sharing computation during a forward pass (see Figure 2.1 (b)), can avoid multi-stage training by training a CNN for both action classification and bounding box regression. Such a detection system is bound to be computationally less expensive than that of [14, 20] and eliminates the need for feature extraction, caching and SVM training. However, this new action detection framework still relies on frame-level action representation to solve for spatio-temporal action localisation. In Section 2.2, we show that a frame-level representation gives a suboptimal solution to the action detection problem, and thus, a video-level action representation is desirable to achieve an optimal solution. In this section, we mainly focus on addressing the problems associated with the existing action detection

[1] For feature extraction, we used 7 Nvidia Titan X GPUs in parallel.

systems [14, 20] such as, poor feature representation, inefficient and expensive network architecture, expensive multi-stage training.

Moreover, previous spatio-temporal human action detection methods [14, 20] use unsupervised region proposal generation algorithms (such as Selective Search [47] and EdgeBoxes [50]) to generate rectangular region hypotheses. As these are unsupervised algorithms, they can not utilise the ground truth action location information provided with a specific dataset. By leveraging deep CNN features and posing the region proposal generation as a supervised machine learning problem, we can generate better quality action region hypotheses with higher recall-to-IoU [29]. Besides, to generate region proposals using these unsupervised methods, each video frame is to be processed individually which is indeed computationally expensive. Whereas, in a supervised approach [29], this expensive per-frame region proposal generation process can be completely eliminated using a fixed set of anchor boxes. Therefore, in order to improve the detection accuracy and reduce the computing cost, it is desirable to model an action detection framework which can (a) generate high quality region proposals by taking advantage of deep representation under a supervised setting and (b) jointly optimise both the region proposal and action detection objectives.

This idea drives us towards designing a novel deep network based action detection framework (Chapter 5) which can generate region proposals by optimising jointly a binary "actionness" [51] classification and a bounding-box regression objectives by exploiting deep features combined with the ground truth class labels and spatial location information associated with each action instance present in the training videos. The "actionness" classification objective assigns high scores to those proposals which are highly likely to contain an action instance (positive training samples) and low scores otherwise, and the box regression objective helps to improve the localisation accuracy of the positive training samples by regressing them towards the corresponding ground truth boxes. Once the region proposals are obtained, they can be sorted as per their actionness scores and the top k proposals then can be used to train a deep network for action classification and bounding-box regression.

We refer to the visual representation mentioned above as *"frame-level"* representation because it encodes only the static appearance of the actor(s) and the scene from a single video frame, but it fails to encode the motion pattern inherently associated with the action.

An illustration of a frame-level deep action representation. Consider Figure 2.2 (a). A cropped image patch is passed as input to a CNN which

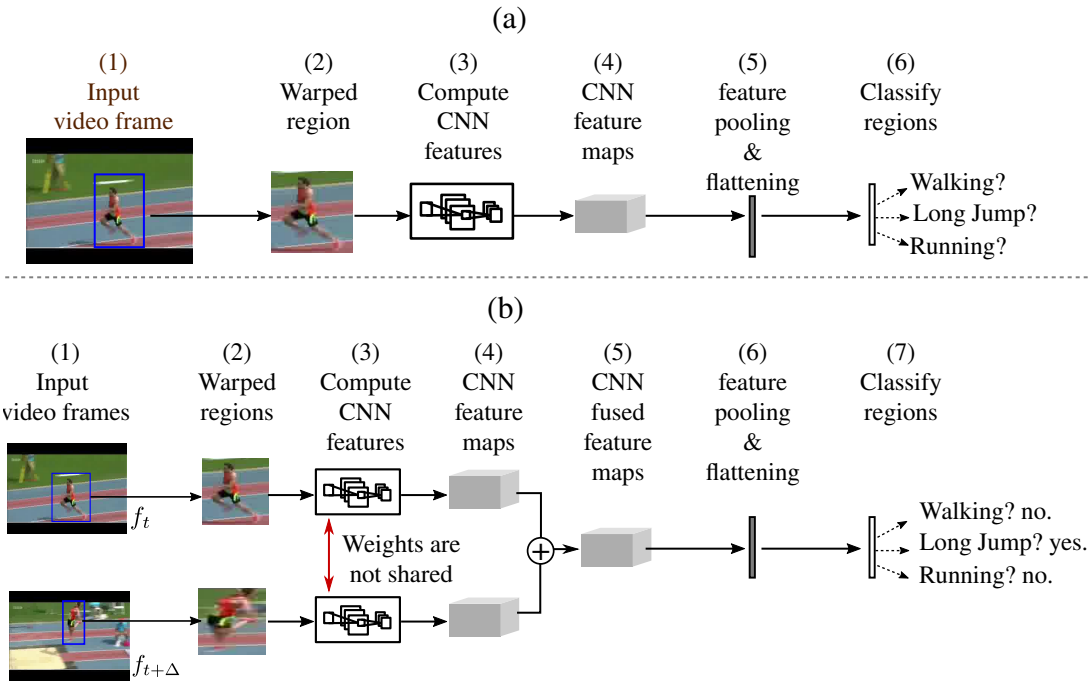


Figure 2.2: Frame-level and video-level representations of actions. ***(a)*** *frame-level representation,* ***(b)*** *video-level representation - approach 1.*

computes deep convolutional features encoding the static appearance of the path. Subsequently, high-level feature encoding is used to train a classifier for action recognition. Note that, this feature encoding encodes only the static appearance cues due to the fact that it belongs to a single video frame. In the following section, we discuss a *"video-level"* representation able to encode both spatial and temporal features of an action instance within a video.

2.2 Deep feature representation of spatio-temporal action instances

This dominant paradigm for action detection [14, 20, 32, 33], however, only provides a *suboptimal* solution to the problem. Indeed, rather than solving for

$$T^* \doteq \arg\max_{T \subset V} \text{score}(T), \tag{2.1}$$

where T is a subset of the input video of duration D associated with an instance of a known action class, they seek partial solutions for each video frame

$$R^*(t) \doteq \arg\max_{R \subset I(t)} \text{score}(R), \tag{2.2}$$

to later compose in a post-processing step partial frame-level solutions into a solution

$$\hat{T} = [R^*(1), ..., R^*(D)] \tag{2.3}$$

to the original problem (Equation 2.1), typically called *action tubes* [2] [14]. By definition,

$$score(\hat{T}) \leq score(T^*) \tag{2.4}$$

as $\hat{T}$ is selected after searching a much smaller (sub)space of solutions compared to the original problem (Equation 2.1). Therefore, such methods are bound to provide suboptimal solutions. Note that, the action classification accuracy of the system solely depends on the tube score ($score(\hat{T})$ or $score(T^*)$), i.e., we assign a class label to a test video with the label predicted for the best scoring tube. Thus, an optimal solution (T^*, see Equation 2.1) is expected to improve the classification accuracy resulting an overall improvement in the detection performance.

More specifically, in the post-processing step frame-level detection bounding boxes are linked in time to build action tubes either by using a Viterbi [14,32,33] or a tracking-based [20] algorithm. This post-processing step is essential as those CNNs do not learn the temporal associations between region proposals belonging to successive video frames (i.e. the inter-frame data associations (Section 1.3)). For instance, an *"archery"* action can be easily identified only from a still video frame due to its salient appearance cues like the presence of bow and arrow. However, actions with similar appearance features such as: *"crawl"*, *"breaststroke"* and *"swim"*, *"laugh"* and *"yawn"*, *"walk"* and *"run"* are hard to discriminate only using still frames as they might be sometimes ambiguous due to their inter-class confusion (Section 1.3), and thus, there is a need to incorporate motion features to achieve more discriminative action representation. To compensate for this and learn the temporal dynamics of human actions, the two-stream architecture makes use of optical flow based CNN feature representation (Section 2.6) to encode the motion pattern of human actions. Although such flow based CNN feature representation helps to improve the action detection accuracy, the architecture of these CNNs have limited temporal scale as the networks operate on either only a pair of consecutive optical flow frames [14, 20, 33] or

[2] Notice, we need to generate tubes specific to actions instead of the actors because, we want our system to detect human actions which consist of both the actor(s) and the contextual information, e.g., in a *"shoot-gun"* action, the gun is a contextual information. Similarly, in a *"climb-stairs"* and *"shoot-bow"* actions, the stairs and the bow/arrow are the contextual objects present alongside the actors.

a stack of 5 flow maps [32]. This frame-level representation is mostly suitable for object detection, but inadequate for action detection where both spatial and temporal localisation are crucial.

Such limitations motivate us to design a new deep learning architecture (Chapter 7) where (a) representations of space-time action instances are learnt from subsets of video frames (i.e. *"video-level"* representations), (b) space-time 3D region proposals (Section 2.3) are exploited for video-level training as opposed to 2D region proposals used for frame-level training and (c) action micro-tubes (Section 2.3), as opposed to frame level bounding box detections, are temporally linked (Section 2.4).

Another drawback with these two-stream based methods [14,20,32,33] is that the network can not learn the pixel-wise correspondences between appearance and motion features as the fusion is performed only at test time [52]. We extend the deep architecture presented in Chapter 7 by leveraging the video-level action representation combined with a train time fusion scheme to fuse appearance and motion cues which allows the network to learn the pixel-wise correspondences between both RGB and flow features (Chapter 8).

2.2.1 Illustrations of video-level action representation

Video-level representation - approach 1. Now, consider Figure 2.2 (b) where a video-level action representation (Chapter 8) is illustrated. Two cropped image patches belong to two successive video frames (f_t and $f_{t+\Delta}$) are processed through two separate CNNs. The output feature maps of these two CNNs are fused using an element-wise *sum* fusion [52], and subsequently the fused feature representation is used to train for action classification. Note here, this feature representation encodes both the appearance and motion cues as the encoding belongs to a pair of successive video frames. Also notice, the two parallel CNNs do not share weights and learn their own set of model parameters (weights) independently during training. In other words, these two parallel streams learn two different representations which are then fused by the element-wise *sum* operation (on the convolutional feature maps) to learn a spatio-temporal encoding of actions.

Video-level representation - approach 2. Another variant of video-level action representation [53] is shown in Figure 2.3. Cropped image patches belong to a set of video frames (f_1, $f_2 \ldots f_{t+\Delta}$) are processed through their corresponding CNNs (weights are shared among these CNNs) and subsequently the output

feature maps are concatenated to obtain a single feature representation.

The main drawback of approach-2 is that the dimensionality of the concatenated feature representation increases linearly with the number of video frames leading to a very high dimensional feature representation in case of longer video subsets. Further, as the weights are shared among CNNs, these might not learn the salient temporal cues independently. Whereas, in approach-1, as the weights are not shared between the two CNNs, we expect them to learn independently the different motion cues at two different time points t and $(t + \Delta)$. Hence, in this work, we use approach 1 for video-level representation.

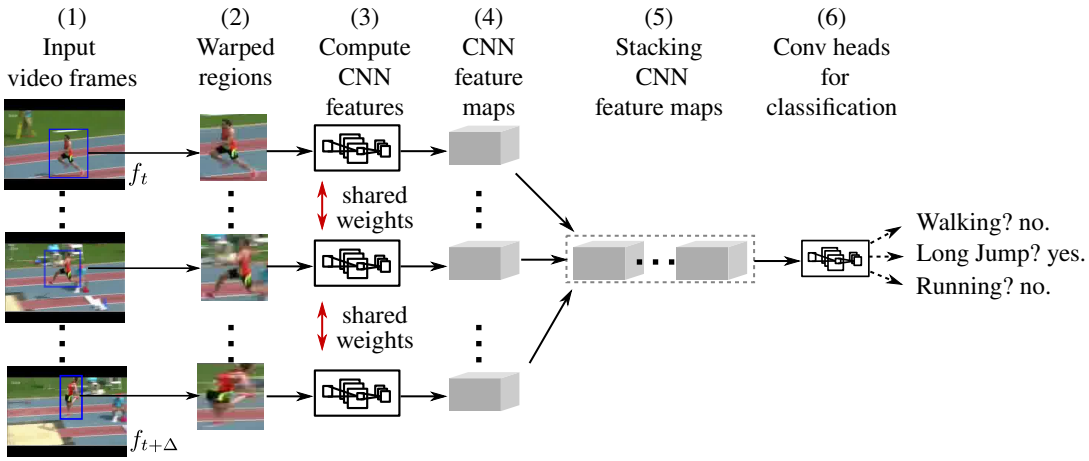


Figure 2.3: Video-level representations of actions - approach 2.

2.3 Deep learning of action micro-tubes using 3D proposal regression

Frame-level region proposals [14, 20, 32, 33] are rectangular region hypotheses used to train a bounding-box regressor for spatial action localisation. Unsupervised region proposal algorithms [47, 50] were heavily exploited [14, 20] to generate action region hypotheses. Due to the fact that this proposal generation algorithms are unsupervised and are disjoint from the overall training optimisation, a region proposal network (RPN) architecture was proposed in [29]. RPN is a fully convolutional neural network which generates region proposals by first initialising the 2D image search space with some predefined anchor boxes, and subsequently, regressing the best matched (positive) anchor boxes towards the ground truth based on the location-specific convolutional features. The upside of RPN are: (a) it is fully supervised and (b) can be integrated into the optimisation process, leading to relatively higher recall-to-IoU [29]. RPN is thus suitable

for generating 2D region proposals for frame-level representation and training. For video-level action representation (Section 2.2) and training, we require action region hypotheses which span both space and time. To this end, we propose a 3D-RPN network (Section 7.3.2) to generate action region hypotheses spanning both space and time by extending the RPN network [29]. Below we explain the concept of *3D region proposal* and *action micro-tube* with an example which helps the readers to understand the action detection frameworks presented in Chapter 7 and Chapter 8.

Consider Figure 2.4 (a). A frame f_t from a *"diving"* video sequence is shown (taken from UCF-101-24 action detection dataset [17]). The ground truth bounding-box is shown in green and a closely matched anchor box (or a 2D region proposal) is shown in red. In frame-level action representation and training, a regression loss is minimised (based on features computed by a CNN) to learn a transformation that maps the anchor box (in red) to the ground truth box (in green) [28].

Now consider Figure 2.4 (b) where a *"diving"* action spans two successive (but necessarily consecutive) video frames f_t and $f_{t+\Delta}$. In this case, a ground truth action **micro-tube** (i.e. a pair of ground truth boxes belongs to frames f_t and $f_{t+\Delta}$ respectively) is shown in green. One of the best matched **3D region proposals** (i.e. a pair of anchor boxes which has high mean overlap with the corresponding ground truth boxes) is shown in red. During video-level training, a transformation is learnt to map the 3D region proposal (red) to the ground truth micro-tube (green).

Lastly, unlike frame-level action detection methods [14, 20, 32, 33], which output detection bounding-boxes at test-time, an action detection model trained on video-level features outputs detection micro-tubes (Chapter 7). Temporally linking detection micro-tubes is faster than linking frame-level detections (for more details, please refer to Section 7.5 and 8.3.2). Figure 2.4 (c) shows the temporal linking of the action micro-tubes extracted during test time.

2.4 Inter-frame data association and temporal detection

In action detection, the inter-frame temporal association or temporal correspondence (of action instances) problem is mostly solved by using graph-based video segmentation methods and shallow features (e.g. dense trajectories [54])

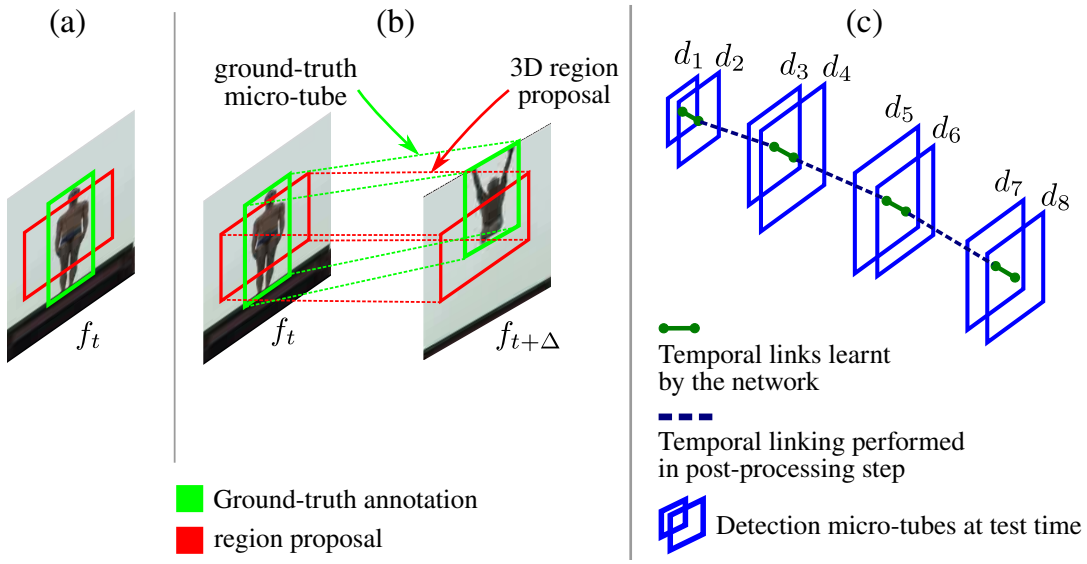


Figure 2.4: Space-time action localisation. ***(a)****: image-level regression - learns a transformation (based on features computed by a CNN) that maps the anchor box (in red) to the ground truth box (in green);* ***(b)****: video-level regression - learns a transformation that maps the 3D region proposal (in red) to the ground truth micro-tube (in green);* ***(c)****: linking detection micro-tubes in time at test time.*

[22, 23, 25, 55]. These approaches rely on highly expensive and time consuming steps: extraction of supervoxels (at diff. levels of segmentation hierarchy) and dense trajectory features. More recently, the temporal linking of frame-level detections (or the data association problem) (Section 1.3) is performed either by solving an optimisation problem using Viterbi algorithm as in [14, 33] or applying tracking-by-detection approach [20]. One highly successful approach is to pose the temporal linking as an optimisation problem [14] in which frame-level detections are linked in time as per the overlaps of their spatial locations and their class specific confidence scores. Another promising solution is the tracking-by-detection approach where, the best frame-level detection is tracked over the entire video. The key factors affecting the performance of a tracking-by-detection approach are: (1) a robust selection criteria for picking the best detections to track and (2) an effective initialisation technique for the tracker [20].

The results obtained after solving the above temporal linking problem is a set of class-specific action tubes (or tracks) which do not explicitly carry any information about the start (initiation) and end (termination) time poins of their respective action instance. To detect the start and end times of each human action instance present in a video, we need to design a model which can perform temporal detection (or localisation) for us 1.3).

A common approach for temporal detection is to apply a sliding window on

action tubes (tracks) [20] at different temporal scales. For instance, Weinzaepfel *et al.* [20] use 13 different temporal scales (e.g. window length of 20, 30, ..., 600). A sliding window is expensive due to a large search space along different temporal scales. The runtime complexity of a sliding window algorithm is $\mathcal{O}(T^2)$ where T is the duration of the video. For longer video sequences, traversing this huge search space is computationally expensive and not suitable for applications requiring online and real-time performance.

A simple and more robust solution is to pose temporal detection as an energy optimisation problem [33] and apply temporal label smoothing (as in Evangelidis *et al.*in [56]) for each action tube individually. By posing it as a label smoothing problem, we reduce the algorithm's runtime complexity from polynomial time $\mathcal{O}(T^2)$ (for sliding window) to linear time $\mathcal{O}(T)$. This allows the design of a more cost effective algorithm suitable for online real-time applications. Evangelidis *et al.* [56] solve for a multi-label smoothing problem, i.e. applying label smoothing to detect the temporal extents of 25 different gestures in videos. Label smoothing is performed based on the 25 frame-level confidence scores. In our case, as we have action tubes, each of which belongs to a particular action category c, we can cast temporal action detection as a binary label smoothing problem. We can thus obtain a temporally trimmed action tube by assigning each detection box (of an action tube) either a class label c or 0 (a 0 denotes a background class) as per their class-specific confidence scores. (Chapter 5).

2.5 Spatio-temporal action instance segmentation

The problem of *"action instance segmentation"* in images can be considered as an intersection of both (a) semantic action segmentation and (b) frame-level action detection [57]. In semantic segmentation, each pixel in a video frame is assigned an action class label. However, these are not instance-aware labels, i.e. different instances of the same action category can not be uniquely identified. Figure 2.5 (a) shows the output of an ideal semantic segmentation method. In this video frame, there are two instances of the *"typing on keyboard"* action class and one instance of the *"entering an office"* action class. Although, semantic segmentation can successfully assign to each pixel its action class label, it fails to assign instance-aware class labels to two different instances of the same class *"typing on keyboard"*. Frame-level action detection does provide both class-

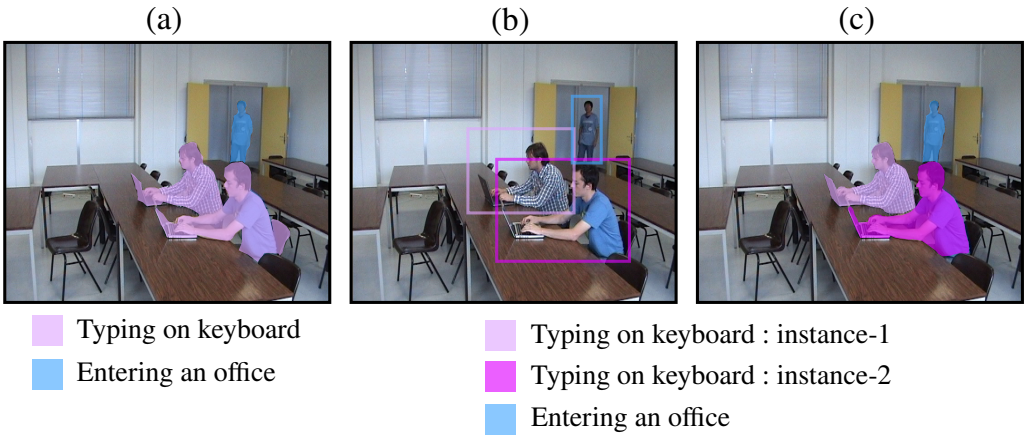


Figure 2.5: Action instance segmentation can be though of as an intersection of semantic segmentation and frame-level action detection. Output of an ideal ***(a)*** *semantic segmentation,* ***(b)*** *frame-level action detection and* ***(c)*** *action instance segmentation methods. Note that, in* ***(a)****, instance-aware class labels are missing. Whereas, in* ***(c)****, the two different* "typing on keyboard" *action instances can be precisely located using pixel-level instance-aware class labels. Each colour of the overlaid masks on the video frames denotes an action class label which is not instance-aware in* ***(a)****, but is indeed instance-aware in* ***(b)*** *and* ***(c)****.*

and instance-aware labels, but labeling is done at a very coarse, bounding-box level. The output of an ideal frame-level action detector is shown in Figure 2.5 (b). Action instance segmentation provides both class- and instance-aware labels at pixel-level (see Figure 2.5 (c)). Unlike semantic segmentation, instance segmentation can uniquely identify instances of the same class. Unlike frame-level action detection, it assigns a label to each pixel instead of each bounding-box.

Although a lot of research initiatives have been taken for action detection [14, 20, 32, 33], yet, we have not noticed any research work in the direction of space-time action instance segmentation. Emerging real-world applications require an all-round approach to the machine understanding of human behaviour which goes beyond the bounding-box level space-time localisation of human actions. For instance, *assume a self-driving car wants to localise a particular instance of a* "pedestrian" *action class among many pedestrians in a crowded scene.* In such scenarios, a spcae-time *"action instance segmentation"* method can enhance the localisation capability of the self-driving car by delineating the pedestrian at finer pixel-level (as opposed to coarse bounding-box level).

Multiple instances of actions/objects can be detected using bounding-boxes; however, applications such as autonomous driving and robot-assisted surgery where depth information is available, a pixel-level segmentation mask can further facilitates a more precise 3D localisation and segmentation [58] beneficial

for tasks such as obstacle avoidance and path planning. Besides, a mask allows a reasoning about occlusion and depth layering, thus, carry more richer information than a bounding-box.

Motivated by the aforesaid advantages of a pixel-level segmentation mask over a detection bounding-box, in Chapter 6, we take a first step towards the design and implementation of a deep learning based framework able to perform space-time action instance segmentation alongside action detection. Just to be clear, the main aim of this thesis is not to address the problem of "instance segmentation", but to propose novel models for action detection, and throughout this work we use only 2D RGB and optical flow frames.

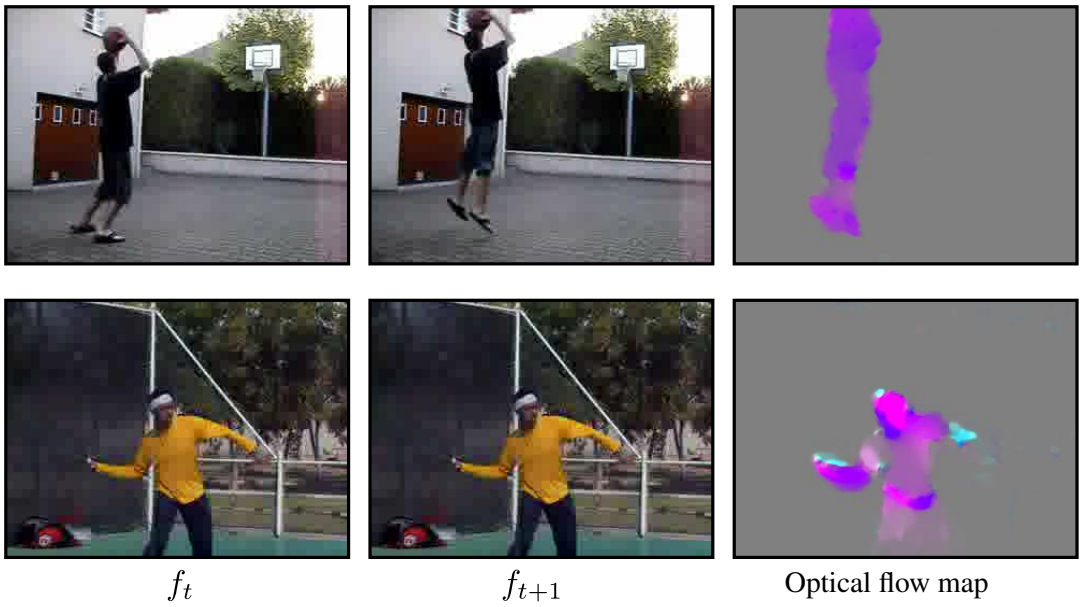


Figure 2.6: Sample optical flow images (maps) computed from pairs of video frames (f_t, f_{t+1}).

2.6 Two-stream hypothesis - fusion of appearance and motion cues

Similar to [14, 27], our detection frameworks (presented in Chapters 5, 6 and 8) are also inspired by the two-streams hypothesis [59] of human vision system. According to this hypothesis, the *apperance* (shape, color and texture) and *motion* (spatial transformations and movement) information are complementary and combining both these cues leads to better understanding of the visual world. In the human brain, the fusion of appearance and motion cues happens naturally. However, for machines, we need to explicitly design an al-

gorithm to combine these two information to achieve better detection accuracy. To simulate the two-streams hypothesis for machines (computing devices), we use two convolutional neural networks (CNNs). The first CNN is used to encode the static appearance of actors and their environment from RGB video frames, we name it as appearance-based CNN. The second CNN is used to capture the pattern of motion of actors and objects (if any) associated with the action (or actions) from optical flow images (or flow maps) (Section 2.7), we name it as motion-based CNN. Figure 2.6 shows some sample flow maps computed from pairs of video frames. We further propose two effective methods for combining the appearance and motion cues: a late fusion technique at test time (Chapter 5) and a CNN feature fusion approach during training (Chapter 8).

2.7 Capturing inter-frame motion pattern using optical flow

In action detection, optical flow signals are heavily exploited to capture the motion dynamics (of human actions) present in videos [14, 20, 33]. The motion patterns present in two consecutive video frames can be captured by computing dense optical flow fields between these two frames [27] (Figure 2.6). In Chapter 5 & 6, our motion-based CNNs operate on training (or test) examples each of which consists of a single flow map. Recently, Peng *et al.* [32] have demonstrated that action representation can be further improved by stacking optical flow signals over multiple video frames (e.g. 5 or 10 frames) which leads to a significant improvement in the detection performance. Motivated by this fact, we train (or test) our motion-based CNN on training (or test) examples each of which composed of stacked flow maps (Chapter 8). We refer to Section A.1 for detials on optical flow map computation.

Chapter 3

Related work

Before we move on to discussing the technical contributions of this thesis, a detailed review of the most relevant work on action classification and detection is in place. We first briefly review the prominent work in action classification in Section 3.1. Then in Section 3.2.1, we outline the recent advances in temporal action detection. Finally, we review the state of the art in spatio-temporal action detection in Section 3.2.2. In the computer vision community, it has been demonstrated that a CNN can learn a better representation by increasing the *depth* of the network i.e. by increasing the number of layers [30]. Therefore, representation learnt by these networks are often referred as *"deep"* representation. In contrast, the classical representation is referred as *"shallow"*.

3.1 Action classification

A plethora of action classification methods have been proposed to recognise human actions in videos. For a detail review, we refer to the recent surveys [60–63]. The major advancements in this field can be attributed to the rapid progress in video representation, starting from shallow representation [42] to the latest generation deep representation [46]. Shallow representation based methods are mostly handcrafted and relatively simple. Whereas, deep convolutional neural networks (CNNs) are built with multiple layers of non-linear feature extractors and are relatively more sophisticated than the standard shallow representation.

3.1.1 Shallow representation

Most of the shallow representation based methods rely on local spatio-temporal features (video descriptors), in which space-time interest points are detected

using either dense fixed grid or a variety of Interest Point Detectors (IPDs) [64–67]. Subsequently, local spatio-temporal features are computed from the pixels around each interest point to describe space-time patches. These descriptors are then transformed into more invariant representations using higher order encodings such as bag-of-visual-words (BoVW) or Fisher vectors. Finally, the encoded feature vectors are used to train classifiers (e.g. SVM, decision forests) for action recognition. As this interest point based local feature representation avoids a strict assumption about an action's global structure, it outperforms global representation where videos have realistic actions captured under uncontrolled settings. The local space-time features are directly computed from raw pixel values, and thus, they minimise the risk of failure due to the use of error-prone processing steps such as long-term tracking, contour/silhouette extraction or background/foreground segmentation. Mostly, local spatio-temporal features were derived from their 2D counterparts: Cuboid [66], 3D-SIFT [68], HoG-HoF [69], Local Trinary Patterns [70], HoG3D [71], extended SURF [67], C2-shape features [72]. For instance, Laptev [65] and Klaser *et al.* [71] extend Harris' cornerness criterion [73] and Histogram of Oriented Gradient (HOG) [74] to videos respectively. Whereas, for video descriptors, Laptev *et al.* [69] compute both HOG and HOF (Histogram of Optical Flow) features.

Unlike still images, the defining feature of video is motion, - an effective way to encode motion is to extract features along trajectories. Thus, a variant of shallow representation is trajectory-based features in which motion in video is represented by long-term point trajectories [75–77]. However, tracking image points over long video sequences is challenging due large displacements, occlusions and above all it is expensive.

To overcome these limitations, Matikainen *et al.* [78] and Wang *et al.* [79] track point coordinates for a relatively shorter duration (e.g. 15 frames) and aggregate these short-term trajectories (also known as tracklets or trajectons) for video representation. Unlike long-term, short-term trajectories are robust to drifting (due to their shorter length) and can be directly computed from optical flow [80, 81]. Space-time patches (video subvolumes) centred at these short-term trajectories are often described via both appearance and motion features. For instance, one of the most successful video descriptors is *dense trajectory* [54] which is formed by combining the HoG-HoF [69] and motion boundary histogram (MBH) [82] descriptors, together with a sequence of optical flow displacement vectors. Unlike local space-time feature based approaches, dense trajectory features do not rely on interest points and extract features from trajectories com-

posed of points from a dense grid. The action recognition performance of these trajectory based approaches can be further improved by discarding the motion caused by the camera movements [83–85]. To this end, camera motion is estimated by matching feature points (e.g. SURF [86]) between video frames.

3.1.2 Human location centric approach

Unlike the shallow representation based methods (Section 3.1.1) in which the action's location information is discarded, some approaches [87,88] actually make use of this location information. Approaches based on human location are often called *"location centric"* or *"figure-centric"* methods. Efros *et al.* [87] proposed a method to recognise human actions at a distance (i.e. at a low resolution) using a combination of both shape and motion (optical flow) features. Action prediction is performed using nearest neighbour techniques. The main limitation is that it requires the actor's location information beforehand to perform action classification. Blank *et al.* [88] utilise the action location information by assuming an action instance as a 3D shape extracted by the human silhouettes from the space-time volume. The downside of their approach is the assumption of a known background. Raptis *et al.* [89] combines ideas from part based models [90,91] with the extraction of a sparse, low-level video representation to solve for action recognition. Jhuang *et al.* [18] find that high-level human pose features substantially improve action recognition.

3.1.3 Deep representation

Recently, the latest generation of deep representation [46] based methods substantially outperformed any shallow representation (Section 3.1.1) based approach in several computer vision tasks such as image classification [30, 92–94], object detection [28] and semantic segmentation [95] in images. These deep convolutional neural networks (CNNs) have also shown impressive results in video classification. 3D convolution can be exploited to encode sequences of video frames for classifying videos [96–98]. Ji *et al.* [96] have proposed a 3D CNN architecture in which sequence of frames are transformed to 3D feature maps by convolving them with 3D kernels over both spatial and temporal dimensions. Karpathy *et al.* [97] have addressed the task of large-scale video classification using 1M videos where different deep network architectures are explored by fusing information over consecutive frames at various levels. Tran *et al.* [98] have used $3 \times 3 \times 3$ kernels for all convolution layers to solve for various vision-based

tasks including action classification, object recognition, scene classification, action similarity labelling.

Donahue *et al.* [99] have leveraged the strengths of deep representation for action classification by combining a CNN with a Long Short-Term Memory recurrent neural network (LSTM) where a chunk of video frames (variable length input) are transformed to visual features by processing them through a 2D CNN, and subsequently, these time-varying visual features are fed into a stack of LSTM units to jointly learn both temporal dynamics and convolutional representation of human actions.

One of the most appreciable work in this direction is the *"two-stream architecture"* proposed by Simonyan and Zisserman [30] in which appearance and motion cues of human actions are encoded using two separate CNNs (streams) by processing RGB and stacked optical flow frames respectively. We adopt this two-stream structure to design the different deep networks proposed in this work. However, we tackle a different problem (i.e. action detection) than action classification. Also, our appearance and motion fusion strategies are entirely different from [30]. Following this pioneering work on the two-stream architecture, Feichtenhofer *et al.* [52] have studied various ways of fusing appearance and motion streams to take the utmost advantage of the spatio-temporal information. They have shown that the new ConvNet architecture based on their findings achieves state-of-the-art results in action classification.

More recently, improvements have been proposed to obtain better spatio-temporal deep representation and to speed-up the action recognition task [100–103]. Arandjelovic *et al.* [100] have explicitly modeled a long-range inhomogeneous video dynamics for accurate recognition of complex human activities. They have brought together both the deep feature representation and VLAD encoding [104] to capture short-term and long-range video dynamics. Motivated by the fact that discriminative actions may present sparsely in a few key video subvolumes, and most of the remaining subvolumes may not contain any action, Zhu *et al.* [101] have proposed a key volume mining deep framework for action recognition. Bilen *et al.* [102] have introduced a concept of dynamic image network which provides a compact video representation using a ranking machine encoding the temporal evolution of video frames. The most computationally expensive step in the two-stream framework [30] is the optical flow computation. Zhang *et al.* [103] have addressed this issue by replacing the optical flow CNN with a motion vector CNN. Their new framework can perform at real-time with a recognition speed of 390.7 fps (frames per second) and shows comparable

classification accuracy to the state-of-the-art.

3.2 Action detection

"Action detection" is also sometimes called as *"action localisation"*. In this section we review various work for spatial, temporal and spatio-temporal action detection.

3.2.1 Temporal action detection

The temporal detection of actions [105, 106] and gestures [107] in temporally untrimmed videos has also recently attracted much interest [56, 108]. In temporal detection, the goal is to locate the optimal subvolume in the 3D video search space. The early approaches to temporal detection rely on sliding-window technique and the emphasis is given on minimising the search complexity [109–111]. Video subsequences with varied temporal extents are uniformly sampled and the one with the maximum classification score is counted as the predicted temporal extent of the action. Unlike spatio-temporal localisation where the search space is extremely large (i.e. it spans over both spatial and temporal dimensions), in temporal detection, the search space is 1-dimensional, and thus, a sliding-window approach is still acceptable.

Gaidon *et al.* [111] propose a more structured representation which is based on decomposing an action as a sequence of atomic action units (actoms). The downside of their approach is the additional cost required to annotate those actoms. Niebles *et al.* [112] exploit temporal patterns of human activities and represent activities as composition of motion segments. They make use of Deformable Part Models (DPM) [91] by inferring temporal scales and anchor points for sub-events of each activity category. Oneata *et al.* [113] reduce the memory and computational cost by proposing an approximation to the normalized Fisher vector which further enables to replace the exhaustive sliding-window search by a more efficient branch-and-bound search [114]. Richard and Gall [115] propose a probabilistic model to solve for temporal action localisation by jointly modelling the temporal segmentation and segment classification tasks. For video segment representation, they use Fisher vector of improved dense trajectories and the the overall objective (segmentation and classification) is maximised using dynamic programming.

CNNs are capable of encoding powerful visual representation whereas, LSTMs

have shown their ability in temporal modeling (or sequence learning), more specifically, in speech recognition [116] and language translation [117,118]. Driven by these recent success of CNNs in visual representation and LSTMs to effectively encode long- and shot-term time-varying signals (speech or video), growing research interests have been noticed to either solve temporal action localisation by purely using CNNs [119] or by combining both CNNs and LSTMs [120, 121]. These work mainly focus on tasks such as: improving representation to better capture motion information at multiple scales, exploring temporal consistency, addressing the difficulties associated with expensive sliding-window technique.

Based on the insight that temporal detection is a process of observing frame glimpses and refining detection hypotheses, Yeung *et al.* [120] combine back-propagation and reinforcement learning to train a network for temporal action detection. Their framework comprises of a CNN for visual representation along with a recurrent neural network (RNN) as an agent and avoids an expensive sliding window scheme to sample action proposals. They introduce a reward mechanism which enables the agent to learn a policy - *"where to look next and when to emit a prediction"*. Yuan *et al.* [121] address the difficulties in multi-resolution sliding window approach by proposing a descriptor (called as PSDF) which captures the multi-resolution context around anchor frames. For temporal consistency, they further combine the PSDF descriptor with a recurrent neural network (RNN). Shou *et al.* [119] exploit the effectiveness of deep CNNs for temporal detection by training three separate networks: (a) proposal, (b) classification and (c) localisation CNNs. They use sliding window to generate action proposals at different temporal scales.

3.2.2 Spatio-temporal action detection

Action cuboid hypotheses and sliding-window based approach. Initial attempts for space-time action detection were based on generating an exhaustive set of cuboid shaped action proposals at plausible spatio-temporal scales using a sliding-window technique [2, 122]. The major drawbacks of these methods are, firstly, the assumption that an action instance within a video has a fixed spatial extent (i.e. an action can be localised using a cuboid video subvolume) is not realistic for unconstrained real-world videos due to the fact that spatial location of an action instance may vary over time. Secondly, the video space is much larger than the image space, and thus, a sliding-window scheme is extremely expensive. For instance, consider a video of size $w \times h \times n$, where $w \times h$ is the spatial and n is the temporal extent, the total number of possible 3D action proposal

is of $\mathcal{O}(w^2h^2n^2)$ [123], it is computationally infeasible to explore such a large search space even for a moderate size video sequence. Lately, Tian *et al.* [124] perform spatio-temporal action localisation by generating 3D action subvolumes using DPM [91], and subsequently, they perform a template matching during test time using a sliding window (i.e. a sliding subvolume) approach.

Human location centric approach. As mentioned earlier in Section 3.1.2, approaches centered at human location are often called as *"location-centric"*. Prest *et al.* [125] use human location information for action detection by first detecting humans and objects in videos, and subsequently, tracking the human-object interactions. Lan *et al.* [126] treat the location of the actor as a latent variable in the Latent Support Vector Machine framework used in the DPM [91] and train it to predict both the location of the actor and the class label of the action simultaneously. Tian *et al.* [124] study the generalization of DPM [91] for action localisation using HoG-3D [71]. Klaser *et al.* [127] propose a human-centric approach where first, spatio-temporal human tracks are obtained using a human detector and a KLT tracker and then, specific actions are classified within the human tracks using a sliding window HOG-3D descriptor [71]. They reported action detection results for only two actions (phoning and standing up). In contrast, in our experiments, we consider 24, 21 and 10 diverse action categories belong to three different human action detection datasets: UCF-101-24 [17], J-HMDB-21 [18] and LIRIS HARL [128] respectively. For action detection, Wang *et al.* [129] use temporal sliding window and the human pose annotations to capture the relations among dynamic-poselets using a sequential skeleton model.

Unsupervised 3D action proposals and shallow representation based approach. Unsupervised 2D region proposal generation algorithms (for object detection in images) [47, 50, 130] have proven their ability to significantly reduce search complexity (over the exhaustive sliding-window technique). Motivated by this success, 3D counterparts of these 2D proposal algorithms are heavily exploited to generate space-time region proposals for action detection [22, 23]. Jain *et al.* [22] and Oneata *et al.* [23] extend the unsupervised Selective Search [47] and Prime object proposal [130] algorithms to their 3D counterparts respectively to generate action tubes (see Section 1.2) or tubelets. They use shallow representation such as dense trajectories [54] and bag-of-visual-words encoding for action tube classification. More specifically, first, video segmentation is performed to generate supervoxels, and then, supervoxels are merged to

form action tubes based on different similarity measures such as color, texture, motion or their size. Similarly, Soomro *et al.* [55] use video segmentation to generate supervoxels (3D action proposals) and use shallow representation for action classification. More specifically, their model learns contextual relations by capturing displacements between supervoxels belong to foreground and background, every supervoxel is encoded using bag-of-visual-words representation on improved dense trajectory [131] feautres. A CRF is used to find the action proposals and one-verus-all SVMs are used to score them. Such 'supervoxels', however, may end up spanning very long time intervals, failing to localise each action instance individually.

The major drawback is that the video segmentation process is highly expensive and not practical for long duration videos or for applications require online and real-time processing. For instance, it takes several minutes to segment a video clip of resolution 400×720 with a temporal length of 55 frames [24]. Another issues with graph based video segmentation are the irregular shape of supervoxels and their highly varying temporal extents [132]. The irregular shape of supervoxels often generates action region hypotheses with lower IoU overlap with ground truth tube leading to poor action localisation performance. Supervoxels with highly varying temporal extents lead to brittle graphs.

Further, generic object proposal generation algorithms [47, 50, 130] are unsupervised in nature and follow a greedy agglomerative clustering of supervoxels, and thus, can not be trained specifically on human action detection datasets. In contrast, we use a fully supervised region proposal generation approach which substantially reduces the computing cost (Chapter 5, Chapter 7 & 8) and can be optimised jointly with the overall action detection objective.

To alleviate the expensive video segmentation process, Van Gemert *et al.* [24] and Yu *et al.* [26] completely bypass the video segmentation step. van Gemert *et al.* [24] generate action tubes by clustering the dense trajectories [54] as per their similarity measures defined by the the HoG, HoF and MBH descriptors. However, since their approach completely rely on dense-trajectory features, i.e. both proposal generation and tube classification is based on dense-trajectories, it may not work on actions characterised by small motions. Further, Yu *et al.* [26] propose an generic action proposal generation method which avoids the expensive video segmentation and shows nearly real-time performance on normal desktop PC. Their methods is based on the notion of "actionness" measure [51] and requires localised training samples. Marian Puscas *et al.* [25] extract unsupervised 3D action proposals (tubes) by leveraging the strengths of both appearance-

based static "objectness" (i.e. Selective Search [47]) and motion (dense trajectories [54]) information. They further refine the tubes by applying transductive learning. Soomro *et al.* [133] recently propose an online method which can predict an actions label and location by observing a relatively smaller portion of the entire video sequence. However, [133] only works on temporally trimmed videos and not in real-time, due to the expensive segmentation method employed.

Two-stream deep representation based approach. Indeed methods which exploit the two-stream deep architecture [27] (Section 2.6) and temporally connect frame-level region proposals [47, 50] for action detection have risen to the forefront of current research. Gkioxari and Malik [14] leverage the deep framework for object detection [28] combined with a two-stream architecture [27] to tackle action detection. Their framework relies on Selective Search proposals [47] and an expensive multi-stage training pipeline as in [28] (Section 2.1). However, as the videos used to evaluate their work only contain one action and are already temporally trimmed (J-HMDB-21 [18]), it is not possible to assess their temporal localisation performance. Weinzaepfel *et al.* [20] also take a similar approach as [14] by adopting a deep object detection framework [28] and a two-stream network structure [27]. They replace the slow Selective Search proposals with relatively faster EdgeBoxes [50] proposals. Further, they use a tracking-by-detection approach based on a novel track-level descriptor called a Spatio-Temporal Motion Histogram. Moreover, [20] achieves temporal trimming using a multi-scale sliding window over each track, making it inefficient for longer video sequences. The common drawbacks of these approaches are: 1) they are computationally expensive and slow due to several factors such as expensive region proposal algorithms, CNN feature extraction and feature caching (Section 2.1); 2) they require multi-stage training i.e. frame-level visual representation is learnt by a CNN and action classification is learnt by a set of one-versus-all SVMs; 3) due to the unsupervised nature of the proposal generation process, the proposal generation task can not be optimised jointly with the overall action detection objective.

In this thesis we improve on both [14, 20] by proposing an elegant and effective solution (Chapter 5). Firstly, we train a deep CNN for proposal generation (Section 5.3.1) on specific action detection datasets in a supervised setting (as opposed to unsupervised proposal algorithms which can not learn "actionness"). Once the CNN is trained, extracting proposals during test time is nearly a cost-free solution [29]. Secondly, we train a CNN to learn frame-level action representation, action classification and proposal box regression (Section 5.3.2)

which further reduces the computing cost substantially. With this new framework, the expensive feature extraction, feature caching and SVM training steps are completely bypassed. 3) We further, replace the expensive sliding-window for temporal detection [20] with an efficient dynamic programming approach (Section 5.3.4). Some of the reviewed approaches [20, 24] could potentially be able to detect co-occurring actions. However, [20] is limited to producing a maximum of two co-occurring detections per class, while [24] does so on the MSRII dataset [122] which only contains three action classes of repetitive nature (clapping, boxing and waving). In contrast, we show evidence that our proposed framework can detect multiple co-occurring detections per class.

Most recently, supervised frame-level action proposal generation and classification have been used by Saha *et al.* [33] and Peng *et al.* [32], via a Faster R-CNN [29] object detector, to generate frame level detections independently for each frame and link them in time in a post-processing step. Unlike [14, 20, 24], current methods [32,33,134] are able to leverage on more faster and elegant deep architectures [29, 135] as compared to [28] for frame level detection. However, tube construction is still tackled separately from region proposal generation.

The new architectures proposed in Chapter 7 & Chapter 8 outputs micro-tubes (i.e. the smallest possible spatio-temporal action region in a video) (Section 2.3) which span across successive frames, and are labelled using a single soft-max score vector, in opposition to [14, 20, 32, 33] which output 2D detection windows at test time. Unlike [14,20,32,33], such models are *end-to-end trainable* and require a *single step of optimisation* per training iteration. This is in contrast to [14,20] which use a multi-stage training strategy mutuated from R-CNN object detection [28] which requires training two CNNs (appearance and optical-flow) independently, plus a battery of SVMs. Compared to [14, 20, 32, 33], which heavily exploit expensive optical flow maps, the action detection framework proposed in Chapter 7 learns spatio-temporal feature encoding directly from raw RGB video frames. Unlike [14,20,32,33], which fuse appearance and motion cues at test time, the action detection pipeline described in Chapter 8 fuses RGB and flow features at training time.

3.3 Action instance segmentation

Action instance segmentation in video is yet an unexplored research area in computer vision. Considerable amount of research has been done to tackle the problem of object instance segmentation in still images [136–141]. However, we

could not find any substantial work addressing the problem of action instance segmentation in videos. Action instance segmentation provides a more elegant and robust solution for accurate human action localisation as compared to a pure detection approach (Section 2.5). Unlike detection, an instance segmentation method can accurately localise actions at a more finer pixel-level by assigning labels to pixels which are both class- and instance-aware.

Early work on object instance segmentation was conducted in [136, 137]. However, instance segmentation has become a more active research topic after the "Simultaneous Detection and Segmentation" (SDS) work of Hariharan *et al.* [138]. They detect all instances of an object category in an image and assign a unique instance-aware label to each instance of that category. Their method is based on the R-CNN object detection pipeline [28]. Several methods [139–141] have extended their work [138]. However, none of those can perform human action instance segmentation in videos and are tailored for object instance segmentation in images. In contrast, in Chapter 6 we propose a two-stream deep representation based framework which exploit both static appearance and motion information from RGB and optical flow signals respectively, and jointly perform action detection and action instance segmentation in video. To the best of our knowledge, we are the first to introduce a deep architecture based action instance segmentation framework.

Chapter 4

Datasets and evaluation metrics

Datasets are the key to validate any machine learning algorithm. Selecting a suitable dataset to evaluate a particular model is thus essential. In this work, we have selected the following two standard benchmarks for action detection: (1) J-HMDB-21 [18] and (2) UCF-101-24 [17]. To support our claim, we would like to mention that these two benchmarks have been used by the previous state-of-the-art action detection approaches [14, 20, 32] to validate their models. In addition to these, we have also used another benchmark LIRIS HARL D2 [128] which has increased level of complexity as compared to J-HMDB-21 and UCF-101-24 (Section 4.1.2).

Outline. This chapter is organized as follows. In Section 4.1, we introduce the datasets used to evaluate the proposed action detection algorithms. We then present the standard evaluation metrics in Section 4.2.

4.1 Datasets

4.1.1 Temporally trimmed videos

Table 4.1: A list of J-HMDB-21 action classes and their corresponding action ids.

Action id	class name	Action id	class name	Action id	class name
1	Brush-hair	8	Pick	15	Shoot-gun
2	Catch	9	Pullup	16	Sit
3	Clap	10	Pour	17	Stand
4	Climb-stairs	11	Push	18	Swing-baseball
5	Golf	12	Run	19	Throw
6	Jump	13	Shoot-ball	20	Walk
7	Kick-ball	14	Shoot-bow	21	Wave

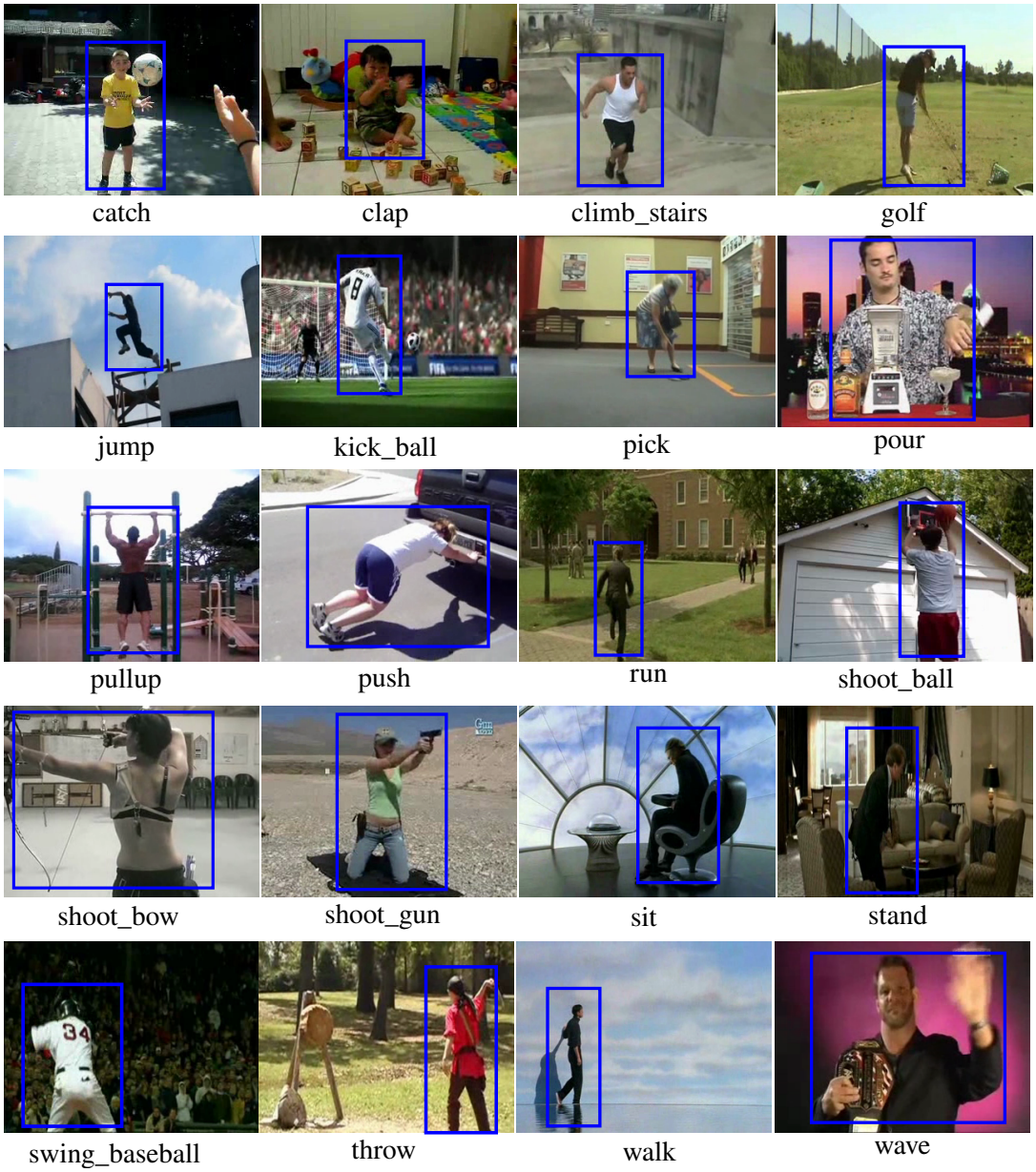


Figure 4.1: J-HMDB-21 sample video frames illustrating the spatial extents of various actions. Blue bounding-boxes show the ground truth spatial locations of action instances.

J-HMDB-21 is a subset of the relatively larger action classification dataset HMDB-51 [4], and is *specifically designed for spatial action detection.* Videos are *temporally trimmed* as per the action's duration, and each sequence contains only *one action instance.* It consists of 928 video sequences and 21 different action categories. A list of 21 action categories is presented in Table 4.1. Sample J-HMDB-21 video frames illustrating the spatial locations of various action classes are depicted in Figure 4.1. A Video duration varies from 15 to 40 frames. Ground-truth bounding boxes for human silhouettes are provided for all

21 classes, and the dataset is divided into 3 train and test splits. For evaluation on J-HMDB-21 we average our results over the 3 splits.

4.1.2 Temporally untrimmed videos

UCF-101-24 dataset

Table 4.2: A list of UCF-101-24 action classes and their corresponding action ids.

Action id	class name	Action id	class name	Action id	class name
1	Basketball	9	GolfSwing	17	Skiing
2	BasketballDunk	10	HorseRiding	18	Skijet
3	Biking	11	IceDancing	19	SoccerJuggling
4	CliffDiving	12	LongJump	20	Surfing
5	CricketBowling	13	PoleVault	21	TennisSwing
6	Diving	14	RopeClimbing	22	TrampolineJumping
7	Fencing	15	SalsaSpin	23	VolleyballSpiking
8	FloorGymnastics	16	SkateBoarding	24	WalkingWithDog

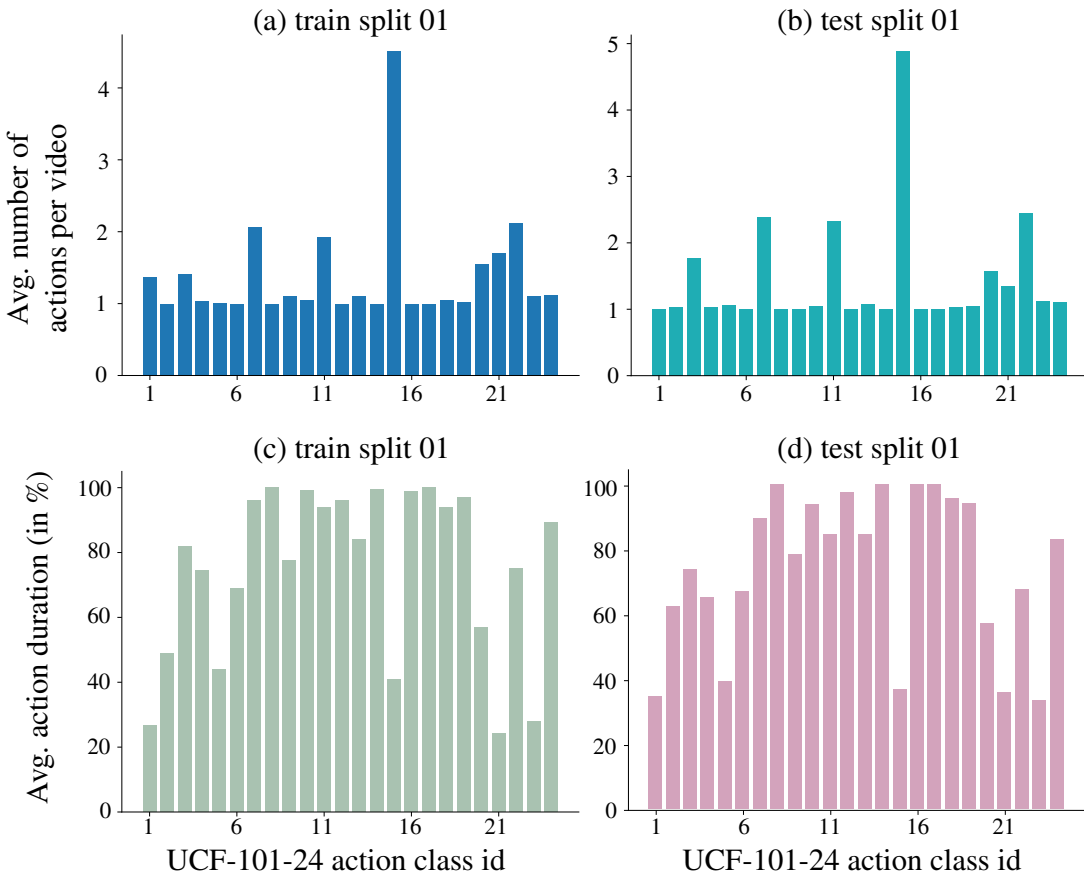


*Figure 4.2: UCF-101-24 dataset statistics. Number of action instances per video: **(a)** train split-01 and **(b)** test split-01; action instance duration: **(c)** train split-01 and **(d)** test split-01; averaged over 24 action classes.*

UCF-101 is one of the largest, most diverse and challenging datasets to date, and contains realistic sequences with a large variation in camera motion, appearance, human pose, scale, viewpoint, clutter and illumination conditions. It is a subset of the larger UCF-101 action classification dataset, and comprises 24 action categories and 3194 (i.e. 2284 videos for train and 910 are for test split-01) [1] videos for which spatio-temporal ground truth annotations are provided. A list of all the 24 action classes and their corresponding action ids are presented in Table 4.2. Although each video only contains a single action category, it may contain multiple action instances of the same action class. We conduct all our experiments using the first split. Compared to J-HMDB-21, the UCF-101 videos are relatively longer and some of them are temporally untrimmed, i.e. action detection is to be performed in both space and time. Figure 4.4 (a) & (b) show histograms of J-HMDB-21 and UCF-101-24 video duration (in number of frames). Note that the THUMOS [106] and ActivityNet [142] datasets are not suitable for spatio-temporal localisation, as they lack bounding box annotation.

Number of actions per video. Number of action instances per video (averaged over 24 action classes) for each action class is plotted in Figure 4.2 **(a)** and **(b)**. Action categories *"biking"*, *"fencing"*, *"ice dancing"*, *"salsa spin"* and *"trampoline jumping"* have videos containing multiple instances. Note that, each video may contain *multiple action instances* happening within different non-overlapping temporal windows or within a same temporal window (i.e. *co-occurring* action instances). Even though videos may contain multiple action instances, they all belong to a same action category i.e. each video is assigned a single class label. In contrast, videos in LIRIS HARL dataset may contain multiple co-occurring action instances belong to *different action categories* and most of them are *temporally untrimmed*.

Class-specific average action duration. We first compute the ratio between an action instance (or action tube) duration and the entire video duration (in number of frames) for all the action instances,

$$\text{ratio} = \frac{\text{instance duration}}{\text{video duration}} \tag{4.1}$$

and then, average those ratios over 24 action classes. These class-specific average action duration (in %) for both train- and test-split01 are shown in Figure 4.2 **(c)**

[1] Out of 3207 videos in UCF-101-24, 3194 videos have correct ground truth annotations available.

Figure 4.3: UCF-101-24 sample video key-frames illustrating the spatio-temporal extents of different actions. Each row represents a UCF-101-24 video sequence belongs to: ***(a)*** *"basketball",* ***(b)*** *"basketball dunk",* ***(c)*** *"cricket bowling",* ***(d)*** *"tennis swing" or* ***(e)*** *"volleyball spiking". Blue bounding-boxes show the ground truth spatial locations of action instances. The ground truth temporal extents of action instances are depicted by the green lines, whereas, video frames which do not contain any actions are denoted by red lines.*

& **(d)** respectively. *"basketball"*, *"cricket bowling"*, *"salsa spin"*, *"tennis swing"* and *"volleyball spiking"* are the most difficult classes for temporal detection due to their relatively shorter action duration. For instances, on an average a *"basketball"* action is only performed in 34% of the entire video sequence. Figure 4.3 shows some sample video key-frames illustrating the sptio-temporal extents of UCF-101-24 action classes.

Difficult and easy classes. Most of the ‘*"basketball"* action instances have a short temporal duration relative to the duration of the video, i.e. the ground truth bounding boxes to spatially localise the actor (in this case the player) are present for a few frames (e.g. see Figure 4.3 (a)). However, the basketball player appears throughout the entire video sequence which makes the temporal detection challenging, that is to detect exactly when the *"basketball"* action initiates and terminates is hard. Similarly, in *"cricket bowling"* class, an actor is present in most part of the video, but the action is annotated within a smaller temporal extent (e.g. see Figure 4.3 (c)). In addition, running (during the *"cricket bowling"* action) is not considered as a part of the action which makes it even more difficult to detect. *"volleyball spiking"* videos contain many potential actors (volleyball players) which are difficult to distinguish (e.g. see Figure 4.3 (e)). On the other hand, *"floor gymnastics"*, *"horse riding"* and *"soccer juggling"* are relatively easier to detect. Possibly, because the average action duration (Section 4.1.2) for these classes are high enough and they are around 100% (see Figure 4.2 **(c)** & **(d)**), that means, most of the videos belong to these classes are temporally trimmed. Further, these classes contain mostly one actor at a time and have salient appearance features. For instance, presence of horse in the *"horse riding"* class.

LIRIS HARL dataset

LIRIS HARL is a human activity detection dataset [128] with 107 training and 58 testing video sequences. The videos in LIRIS HARL have relatively longer duration than the videos in UCF-101-24 and J-HMDB-21 dataset. Histograms of video duration (in number of frames) is presented in Figure 4.4. Note that, majority of LIRIS HARL videos have duration ranges between $150-400$ frames, whereas UCF-101-24 and J-HMDB-21 have relatively shorter video clips, most of them have duration between $75-275$ and $21-41$ respectively.

The dataset was created for an activity detection competition in which 70 teams registered. The large number of activity classes for detection compared to previous datasets [2, 127] and its difficulty meant that only two teams [143, 144] submitted results, to which we compare our results (Section 5.4 & 6.4). The LIRIS dataset is complex because it contains image sequences containing multiple activities annotated in space and time, some of which occur simultaneously. Moreover, it contains scenes where relevant human activities take place amidst other irrelevant human motion (i.e., other people performing irrelevant actions). The LIRIS dataset contains 10 activity categories, a full list of human activity

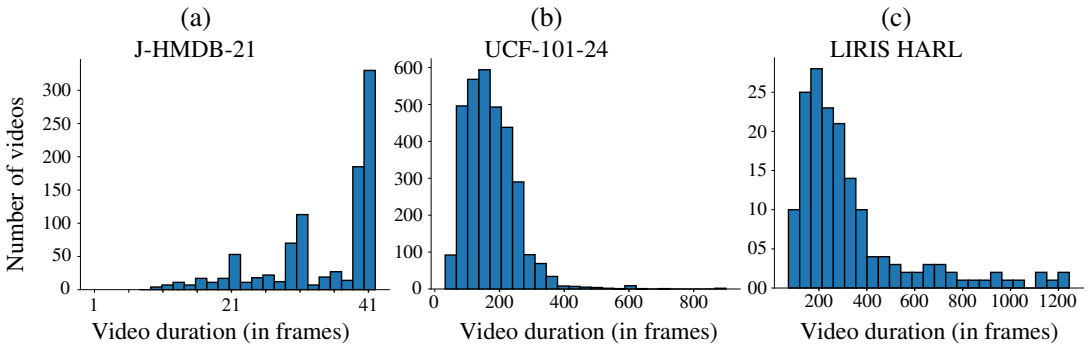


Figure 4.4: Histogram of video duration (in number of frames): ***(a)*** *J-HMDB-21,* ***(b)*** *UCF-101-24 and* ***(c)*** *LIRIS HARL dataset.*

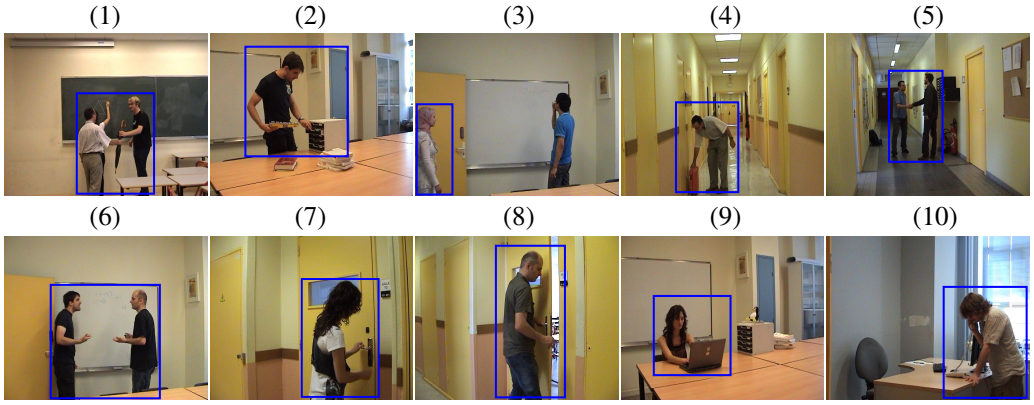


Figure 4.5: Sample LIRIS HARL video frames representing short and long duration activity categories. Short duration classes: ***(1)*** *Give object to person,* ***(2)*** *Put/take object into/from box/desk,* ***(3)*** *Enter/leave room no unlocking,* ***(4)*** *Leave baggage unattended,* ***(5)*** *Handshaking. Long duration classes:* ***(6)*** *Discussion,* ***(7)*** *Try enter room unsuccessfully,* ***(8)*** *Unlock enter/leave room,* ***(9)*** *Typing on keyboard,* ***(10)*** *Telephone conversation. Coloured bounding-boxes denote the ground truth spatial locations of activities.*

Table 4.3: LIRIS HARL dataset - list of human activity categories and their corresponding activity class ids.

Activity Class Id	**Activity class Name**
1	Discussion
2	Give object to person
3	Put/take object into/from box/desk
4	Enter/leave room no unlocking
5	Try enter room unsuccessfully
6	Unlock enter/leave room
7	Leave baggage unattended
8	Handshaking
9	Typing on keyboard
10	Telephone conversation

categories and their corresponding class ids is presented in Table 4.3. In particular, we used the D2 sequences shot with a Sony camcorder with a resolution of

720 × 576, and captured at 25 frames per second.

UCF-101-24 and J-HMDB-21 videos contain atomic actions, whereas, LIRIS HARL videos contain complex activities (collection of atomic actions) which include human-to-human interactions, human-to-object and human-object-human interactions, for example, *"discussion of two or several people"* (Figure 4.5 (6)), *"a person unlocks a door and enters the room"* (Figure 4.5 (8)) and *"a person gives an object to another person"* (Figure 4.5 (1)). Further, LIRIS HARL videos are temporally untrimmed with longer duration (Figure 4.4) which altogether make the temporal association and localisation (see Section 1.3) problems even harder. In the following subsections, we discuss those properties of LIRIS HARL dataset which make it one of the most challenging human activity detection datasets to date.

Class-specific activity duration. The plots in Figure 4.6 (a) & (b) show the class-specific average activity duration (in %) for both train and test sets. We compute the average activity duration similarly as in (Section 4.1.2). Note that,

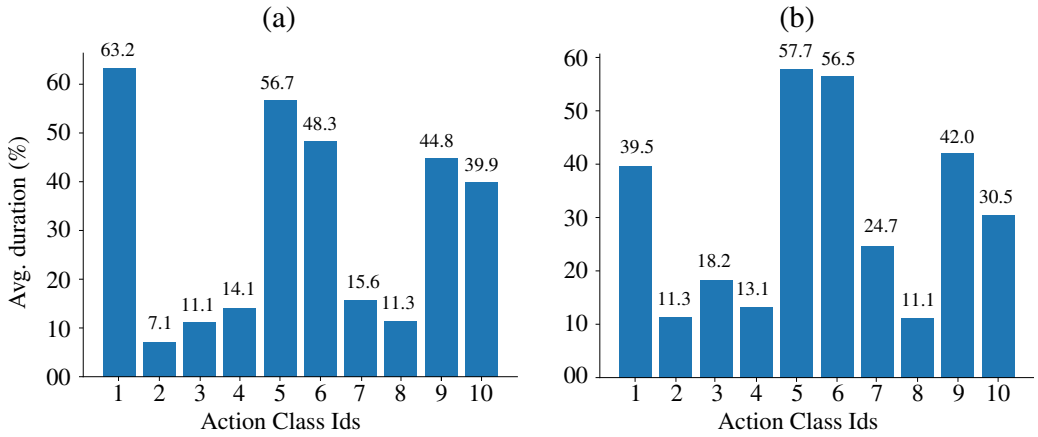


Figure 4.6: Class-specific average activity durations (%), LIRIS HARL dataset: ***(a)*** *trainset and* ***(b)*** *testset.*

unlike J-HMDB-21 and UCF-101-21 which are either fully temporally trimmed or partially untrimmed datasets, LIRIS HARL is a fully temporally untrimmed dataset (see plots (a) & (b) in Figure 4.6), and thus, action detection becomes more challenging. We have shorter duration activities such as *"give object to person"*, *"put/take object into/from box/desk"*, *"enter/leave room no unlocking"*, *"leave baggage unattended"* and *"handshaking"* (Figure 4.5 (1) to (5)), whereas, the remaining activity categories have relatively longer average duration, and they are *"discussion"*, *"try enter room unsuccessfully"*, *"unlock enter/leave room"*, *"typing on keyboard"*, and *"telephone conversation"* (Figure 4.5

(6) to (10)).

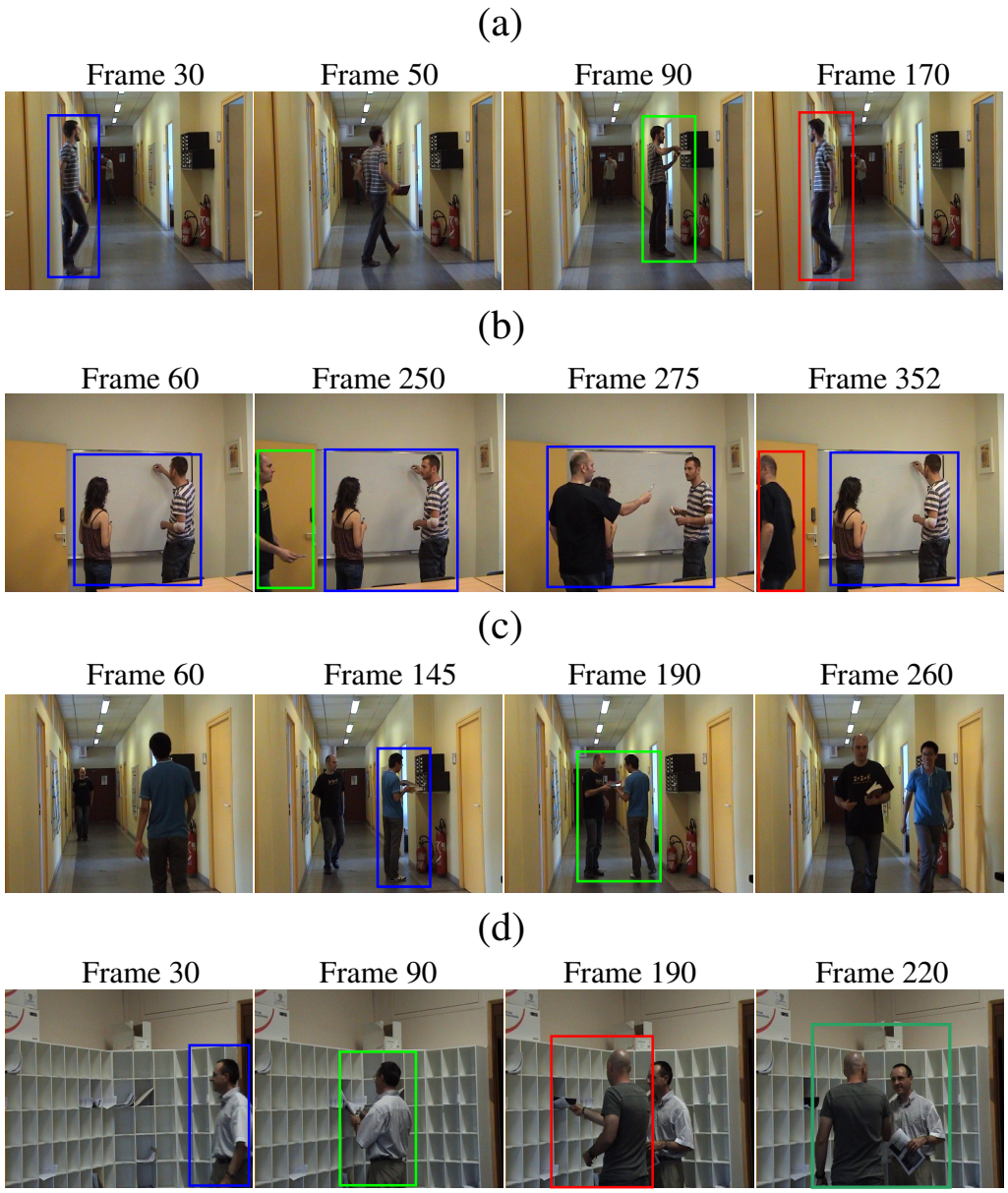


Figure 4.7: Sample LIRIS HARL videos (only the key frames are shown) where multi-label detection is required: ***(a)*** *a person "leaves a room" (FNo. 30), "put an object into a box" (FNo. 90) and then "enters the room" again (FNo.170);* ***(b)*** *two persons engage in "discussion" (FNo. 60), a 3rd person "enters the room" (FNo. 250) and "gives an object to a person" (FNo. 275) then "leaves the room".* ***(c)*** *a person "takes an object from a box" (FNo. 145) and "gives that object to another person" (FNo. 190).* ***(c)*** *a person "enters a room" (FNo. 30) and "puts an object into a box" (FNo. 90), another person also "puts an object into a box" (FNo. 190) and then they "handshake" (FNo. 220). Colours representing different activity instances.*

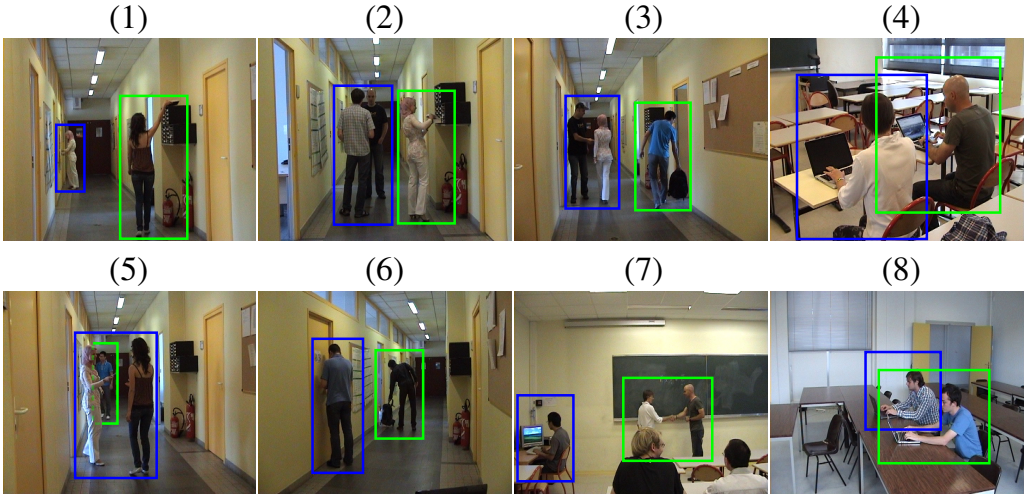


Figure 4.8: Sample LIRIS HARL video frames showing co-occurring activities. ***(1)*** *unlock enter/leave room (blue) and Put/take object into/from box/desk (green),* ***(2)*** *discussion (blue) and put/take object into/from box/desk (green),* ***(3)*** *handshaking (blue) and leave baggage unattended (green),* ***(4)*** *two instances of "typing on keyboard" activity,* ***(5)*** *give object to person (blue) and enter/leave room no unlocking (green),* ***(6)*** *unlock enter/leave room (blue) and leave baggage unattended (green),* ***(7)*** *typing on keyboard (blue) and handshaking (green),* ***(8)*** *two instances of "typing on keyboard" activity.*

Multi-label videos. In UCF-101-24 and J-HMDB-21 dataset, each video is assigned a single class label and actors and scenes belong to different action categories mostly carry discriminative appearance and motion features. For instances, a *"basketball"* action carry different appearance and motion information regarding the player and the background scene when compared to that of a *"fencing"* action. In contrast, in LIRIS HARL dataset, same actors (in a same scene) can perform multiple activities within a single video sequence, e.g. a person *"talks over a phone"* while *"typing on a keyboard"*, two persons *"handshake"* and then start a *"discussion"*, a person *"gives an object to another person"* while they are *"discussing"*. In Figure 4.7, we show sample LIRIS HARL videos (only the key frames are shown) where a group of actors (or a single actor) involved in different activities, some of them happen at the same time. Sample video frames containing *multi-label co-occurring activities* are shown in Figure 4.8. In such real-world scenarios (where multi-label detection is required), activity detection becomes more challenging due to several vision related problems including inter-class similarity, intra-class confusion (see Section 1.3).

Table 4.4: A typical confusion matrix for an action classification task with $C = 3$ classes. GTL: ground truth label; PL: predicted label.

		PL			
		Walk	Jump	Run	Total
GTL	Walk	m_{11}	m_{12}	m_{13}	$m_{11} + m_{12} + m_{13}$
	Jump	m_{21}	m_{22}	m_{23}	$m_{21} + m_{22} + m_{23}$
	Run	m_{31}	m_{32}	m_{33}	$m_{31} + m_{32} + m_{33}$
	Total	$m_{11} + m_{21} + m_{31}$	$m_{12} + m_{22} + m_{32}$	$m_{13} + m_{23} + m_{33}$	N

4.2 Evaluation metrics

4.2.1 Accuracy

In this section, we explain the accuracy measure used to evaluate the action classification performance in Chapter 5 and 6.

The classification Accuracy (Acc) is calculated as:

$$Acc = \frac{\#(\text{correctly classified test video clips})}{\#(\text{total test video clips})} \tag{4.2}$$

Gnerally, for a multi-class classification problem with c $\in \{1, \ldots, C\}$ classes, and i $\in \{1, \ldots, N\}$ testing examples, a $C \times C$ confusion matrix $\boldsymbol{M}$ is first generated by incrementing the element of the matrix at location $\boldsymbol{M}[\text{gt(i)}, \text{pred(i)}]$, for each pair of ground truth and predicted labels. Table 4.4 illustrates a typical confusion matrix. In this case, the accuracy is computed using the following formula:

$$Acc = \frac{\text{the trace of } \boldsymbol{M}}{\text{the total sum of its elements}} = \frac{m_{11} + m_{22} + m_{33}}{N}, \tag{4.3}$$

where m_{11}, m_{22} and m_{33} are the diagonal elements of the 3×3 confusion matrix shown in Table. 4.4.

4.2.2 Precision, recall and F1 measure

In this section, we introduce the precision, recall and F measures which are the prerequisites for understanding the *mean average precision* (mAP) (Chapter 5, 7 & 8) and the LIRIS HARL (Chapter 6) evaluation metrics used to evaluate the proposed action detection models in this thesis.

The two most basic measures which are frequently used to evaluate unranked retrieval results are: (1) precision and (2) recall [145]. Precision (P) is the

fraction of retrieved documents that are relevant

$$\begin{aligned} P &= \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} \\ &= \frac{\#(\text{correctly found action instances})}{\#(\text{number of found action instances})} \end{aligned} \tag{4.4}$$

Recall (R) is the fraction of relevant documents that are retrieved

$$\begin{aligned} R &= \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} \\ &= \frac{\#(\text{correctly found action instances})}{\#(\text{action instances in ground truth})} \end{aligned} \tag{4.5}$$

The notion of precision and recall can be explained using the following contingency table

Table 4.5: Contingency table.

	Relevant	Nonrelevant
Retrieved	true positives (tp)	false positives (fp)
Not retrieved	false negatives (fn)	true negatives (tn)

Thus,

$$P = \frac{tp}{(tp + fp)} \qquad R = \frac{tp}{(tp + fn)} \tag{4.6}$$

$F1$ measure is a single-number that trades off precision and recall and is the weighted harmonic mean of precision and recall. It is found by weighting the 'Recall' and 'Precision' of a classifier equally and is calculated as the ratio:

$$F1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}. \tag{4.7}$$

Although, $F1$ score is a single number measure, it is used to evaluate unranked retrieval results. In the following section we explain the mean Average Precision (mAP) which is a popular single-number measure widely used to evaluate ranked retrieval results.

4.2.3 mean Average Precision (mAP)

The mean average precision (mAP) is a popular single-number measure for comparing search algorithms. Nevertheless, it has been widely used to benchmark detection problems in the action detection literature. Note that the Area Under the Curve (AUC) measure on J-HMDB-21 reported by [14] is sensitive to negative detections, as AUC increases when adding many easy negatives [2], whereas mAP is not affected by easy negatives [20].

In order to calculate the mAP, first we need to compute the average precision (AP) for each class present in the given dataset, and then compute the mean across all the classes. The key here is to compute the class-specific APs. Originally, the AP measure was first used in the Information Retrieval (IR) domain to evaluate the ranked retrieval results of a search algorithm, e.g., a Google style search engine [146]. Unlike precision, recall and the F measure (see Section 4.2.2) which do not take into account the ordering or ranking of the retrieved results (i.e. documents or detections), the AP measure does indeed consider the ranking of the results, and thus, suitable for evaluating ranked retrieval results. In action detection, the final output is a set of detections (either a bounding-box or an action tube) each associated with C classification scores (confidence level), where C is the number of classes. The provision of such confidence level allows the detections to be ranked. Thus, we can consider the output of a detector, i.e., a set of class-specific detections as retrieval results which are ranked as per their confidence level. In the following section, we briefly explain how to compute the AP in the context of action detection problem.

Average Precision (AP)

As evaluation metrics we use both: (1) *frame-AP* (the average precision of detections at the frame level) as in [14, 32]; (2) *video-AP* (the average precision of detection at video level) as in [14, 20, 32, 33]. Average precision (AP) is a single number which is strictly equal to the precision averaged over all values of recall, equates to computing the area under the precision-recall curve (see Figure 4.9). In the popular PASCAL VOC challenge [147], up until 2009 11-point interpolated AP [145] was used to evaluate both classification and detection tasks. However, from 2010 onwards the method of computing AP changed to use all data points. The downside of the interpolated AP is that the evaluation is too crude to discriminate between different methods at low AP [148]. The 11-point interpolated

[2] Negatives which have lower detection confidence than all positives.

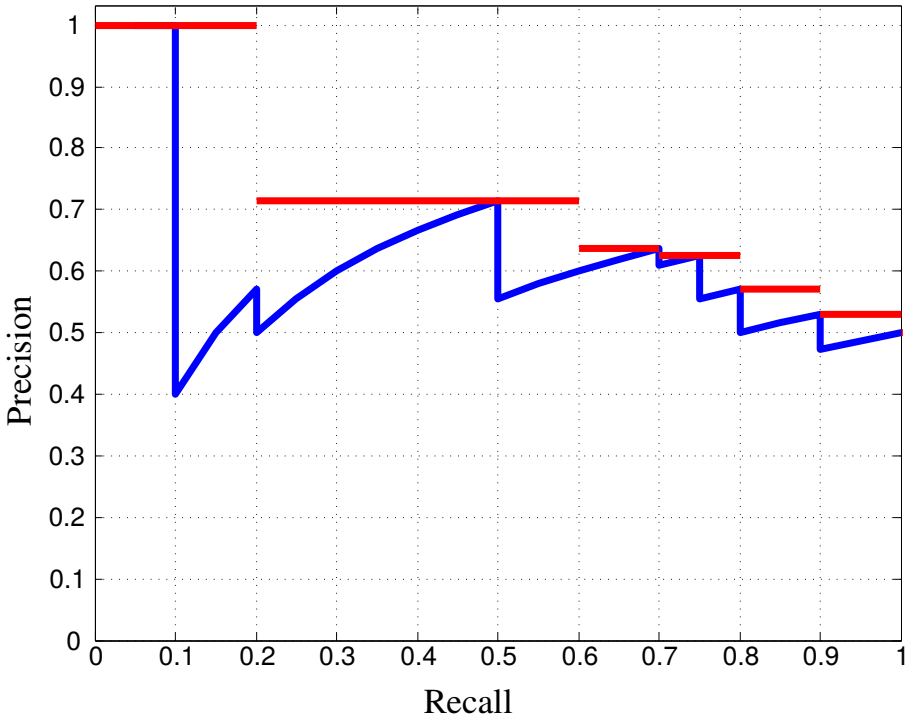


Figure 4.9: An example of a precision-recall curve illustrates the difference between the strict average precision (area under blue curve), and the 11-point interpolated average precision (area under red curve). Example taken from [149].

AP in most instances overestimates the area under the precision recall curve as illustrated by the red lines in Figure 4.9. Following the convention, in this work we use all data points (i.e. the strict average precision) while computing class-specific APs.

In case of action classification, the correctness of a class prediction depends only on whether a video contains an action instance of that class or not. However, for action detection a decision must be made on whether a prediction is correct or not based on the following two criteria :- (1) the predicted class label and (2) the predicted spatial (in case of frame-AP) or spatio-temporal (in case of video-AP) location of the action instance. To this end, detections are assigned to ground truth action instances and judged to be true or false positives (Section 4.2.2) by measuring the *intersection over union* (IoU) overlap (Section 4.2.4) between the predicted detection and the ground-truth instance.

A pseudocode for computing class-specific APs is provided in 4.1 which is adapted from the popular PASCAL VOC development kit code [150]. This is applicable for both frame- and video-based AP computation. Note that, by saying a "detection" here we refer a predicted "action tube" while computing video-AP, whereas, a predicted "bounding-box" in case of frame-AP computa-

tion. Likewise, by saying a "ground truth annotation" we refer a ground truth "action tube" (video-AP) or a ground truth bounding-box (frame-AP). To calculate the AP, first we need to define what are the true positives (tp), false positives (fp), true negatives (tn), and false negatives (fn). In the setting of action detection we can define them as:

- tp: are the detections having same predicted class labels as the ground truth annotation labels, and having IoUs with the ground truth annotations above δ (operation 21);
- fp: are the detections having different predicted class labels from ground truth annotation labels or having IoUs with the ground truth annotations below δ (operation 26); also multiple detections of the same ground truth action instance are considered false positives, e.g. 5 detections of a single ground truth instance counted as 1 true positive and 4 false positives (operation 24);
- tn: there are no true negatives in this case as each frame (in case of frame-AP) or video (in case of video-AP) is expected to contain at least one detection (i.e. a bounding-box or an action tube);
- fn: ground truth annotations with no matching detection are false negatives i.e. those frames or videos where the method fails to produce any detection.

Here, δ is the IoU threshold which is termed as *iout* in the pseudocode. With the above definitions, we now start computing the AP for a specific class c. First, we sort the detections (predicted by the detection algorithm) by decreasing order as per their classification scores for class c (operation 3). After sorting, the detection list becomes an ordered or ranked retrieval results where the first detection belongs to the top rank (i.e. $r = 1$), the second belongs to $r = 2$ and so on up to N-th detection which belongs to the lowest rank, where N is the total number of detections (it is denoted by nd in the pseudocode). Next, we loop over this ordered list of detections and assign each d-th detection to its best matched ground truth if there exist any (operation 5 to 26). Note, the best matching is performed based on the IoU score (Section 4.2.4, operation 16) between the detection and the corresponding ground truth. Also notice, while calculating frame-AP we compute the spatial IoU, whereas, we compute the spatio-temporal IoU during video-AP calculation. During the above assignments of the detections to their best matched ground truths, we simultaneously generate the true positives (tp) and false positives (fp) binary array lists (operation 21, 24, 26). Once the entire detection list has been traversed, we then compute the precision and recall for each rank $r = 1, 2, ..., R$ (operation 27 to 29). Finally, we compute the strict average precision (operation 30) that is the

area under the precision-recall curve.

Implementation note. At the time of testing, for a input video (or frame in case of frame-AP) we may get detections belonging to a different action class, e.g., for a *"VolleyballSpiking"* test video, we may get detections belonging to *"Basketball"* action class due to the inter-class confusion problem (see Section 1.3). These detections are termed as *false positives* (Section 4.2.2) as they are nonrelevant but are retrieved, i.e., they are actually negatives but are predicted to be positives. Now, when we compute the AP for *"Basketball"* class, we need to make sure that these detections are counted as *false positives*.

4.2.4 Intersection over Union (IoU)

In this section we explain the spatial IoU and spatio-temporal IoU computation. The spatial IoU is used to compute the frame-AP, and for video-AP computation we use the spatio-temporal IoU.

Spatial IoU. We compute the spatial IoU (i.e. the area of overlap) between a predicted and a ground truth bounding box, b_p and b_g, using the following formula:

$$IoU_s = \frac{area(b_p \cap b_g)}{area(b_p \cup b_g)} \tag{4.8}$$

where $(b_p \cap b_g)$ denotes the intersection of the predicted and ground truth bounding boxes and $(b_p \cup b_g)$ their union.

Spatio-temporal IoU. We compute the spatio-temporal IoU between a predicted and a ground truth action tube, at_p and at_g, using the following formula proposed by [20]:

$$IoU_{st} = IoU_t \times IoU_s^{mean} \tag{4.9}$$

where, IoU_t is the temporal IoU between the at_p and at_g, and we define it as follows. A predicted action tube has its own beginning and ending time points: t_p^b, t_p^e. Likewise, a ground truth action tube has t_g^b, t_g^e. Thus, we compute the temporal IoU:

$$IoU_t = \frac{T_{intersect}}{T_{union}} = \frac{max(t_g^b, t_p^b) - min(t_g^e, t_p^e)}{min(t_g^b, t_p^b) - max(t_g^e, t_p^e)} \tag{4.10}$$

Next, for each t within the temporal intersection range $max(t_g^b, t_p^b)$ to $min(t_g^e, t_p^e)$, we compute the spatial IoU between the predicted and ground truth bounding boxes, b_p and b_g, and then take the mean of all the spatial IoU values. We denote it as IoU_s^{mean}, see the second term in Eq. 4.9.

4.2.5 LIRIS HARL evaluation metrics

Firstly, any detected action tube is assigned to the closest ground truth tube, based on a normalised measure of overlap over all its frames. Secondly, a detected action tube is accepted as positive if detected and ground truth tubes have the same class, and:

- there is sufficient overlap with respect to thresholds on *"spatial pixel-wise recall"* t_{sr}, and *"temporal frame-wise recall"* t_{tr}, and

- the excess detected space and duration are sufficiently small (refer [151] for more details) with respect to thresholds for *"spatial pixel-wise precision"* t_{sp}, and *"temporal frame-wise precision"* t_{tp}.

Once the four thresholds t_{sr}, t_{tr}, t_{sp} and t_{tp} are fixed, recall, precision and F1-score may be calculated in the usual way as explained in Section 4.2.2.

A final performance measure may be obtained by integrating the F1-score over the range of possible threshold values [151]. Four integrated F1-score values $(I_{sr}, I_{sp}, I_{tr}, I_{tp})$ are first calculated by varying one threshold while setting the others to a small value ($\eta = 0.1$). Then, an overall score is obtained by averaging the four values:

$$\text{Integrated Performance} = \frac{I_{sr} + I_{sp} + I_{tr} + I_{tp}}{4} \tag{4.11}$$

which is independent from arbitrary thresholds on spatial or temporal overlap. We refer readers to [151] for more details on LIRIS HARL's evaluation metrics. The LIRIS HARL evaluation metric [151] requires hyper-parameter optimisation, which is a kind of overhead to the whole evaluation process. In contrast, mAP does not require any hyper-parameter optimisation, and thus, most suitable for measuring performance of spatio-temporal action detection accuracy.

4.3 Chapter wise evaluation metrics

In this section, we use a MATLAB like notation to represent a range of values, e.g., we use $[0.1 : 0.1 : 0.5]$ to denote a range $\{0.1, 0.2, 0.3, 0.4, 0.5\}$, where out of the three values in $[0.1 : 0.1 : 0.5]$, the first and the last values are the start and end of the range, and the middle one is the step value. We refer the readers to Section 4.2.4 for spatial and spatio-temporal IoU, and Section 4.2.3 for frame- and video-mAP metrics. In order to evaluate our action detection approaches, we select the following 3 action detection benchmarks: (a) J-HMDB-21 [18], (b) UCF-101-24 [17], and (c) LIRIS HARL D2 [128]. The rationale behind selecting these datasets can be found in Section 4.1. We use the J-HMDB-21 and UCF-101-24 datasets in Chapter 5, 7 and 8, whereas, the LIRIS HARL dataset is used in Chapter 5 and 6. To compare the action detection results of our approaches with the state-of-the-art mehtods, we use the standard spatio-temporal IoU threshold $\delta = 0.2$ for UCF-101-24 and $\delta = 0.5$ for J-HMDB-21. For frame-mAP comparison, we set the spatial IoU threshold $\delta = 0.5$ for both J-HMDB-21 and UCF-101-24 datasets.

In Chapter 5, we use the video-mAP metric to evaluate the proposed action detection approach on the J-HMDB-21, UCF-101-24 and LIRIS HARL datasets (Section 4.1). We are the first to extensively evaluate our method across various spatio-temporal IoU threshold (δ) range, i.e., for J-HMDB-21 we use $\delta = [0.1 : 0.1 : 0.7]$, for UCF-101-24 $\delta = [0.1 : 0.1 : 0.6]$ and LIRIS HARL $\delta = [0.1 : 0.1 : 0.5]$. We report a video-mAP and integrated F1-Score [151] for the LIRIS HARL dataset at spatio-temporal IoU threshold of $\delta = 0.1$. In Chapter 5 and 6, we also evaluate the qualitative and quantitative performance of our approaches on the LIRIS-HARL dataset using the evaluation metric [151] proposed for the LIRIS HARL competition [128].

In Chapter 7, we compute the video-mAPs on J-HMDB-21 dataset at 5 different spatio-temporal IoU thresholds $\delta = [0.1 : 0.1 : 0.5]$, whereas for UCF-101-24, we compute the video-mAPs at 3 different thresholds $\delta = [0.1 : 0.1 : 0.3]$. In Chapter 8, we compute the video-mAPs on J-HMDB-21 and UCF-101-24 datasets at 5 different spatio-temporal IoU thresholds$\delta = [0.1 : 0.1 : 0.5]$. In Chapter 7 and 8, for UCF-101-24 dataset we also compute the video-mAPs at 10 different spatio-temporal IoU thresholds $\delta = [0.5 : 0.05 : 0.95]$, and then we take the average of these mAPs which we denote as $[0.5 : 0.95]$. The rationale behind this is to check how good the quality of the detections are. An ideal action detector is expected to predict action tubes which have spatio-temporal

IoU = 1.0 i.e., the predicted tubes have exactly the same spatial and temporal locations as the ground-truths. Thus, computing the video-mAPs at higher IoU thresholds allows us to assess the qualtiy of the detections.

Algorithm 4.1 Compute the average precision for a specific action class c.

Inputs:

$cls \leftarrow$ action class name; $gts \leftarrow$ list of ground truths belong to c
$npos = gts.length \leftarrow$ no. of ground truths
$dets \leftarrow$ list of detections belong to c
$keys \leftarrow$ keep record of the video- or frame-id for each detection in $dets$
$nd = dets.length \leftarrow$ no. of detections; $iout \leftarrow$ IoU threshold

Initialise:

$gtObjs = null \leftarrow$ ground truth tubes (video-AP) or bounding-boxes (frame-AP)
$det = null \leftarrow$ a detection tube (video-AP) or bounding-box (frame-AP)
$IoU(.,.) \leftarrow$ compute spatio-temporal IoU (video-AP) or spatial IoU (frame-AP)

function ComputeAvgPrecision(cls, gts, npos, dets, keys, nd, iout)
  if $nd > 0$ **then**
    sort the detections $dets$ by decreasing confidence
    sort the $keys$ as per the sorted detection indices
    // the following code assign the d-th detection
    // to its best matched ground truth if there is any
    for $d = 1$ to nd **do**
      $isfp = False$
      if not $(dets[d].cls == cls)$ **then** // if fp (see note 4.2.3)
        $isfp = True$
      $ovmax = -inf$ // $-inf \leftarrow$ minus infinity
      if not $(isfp == True)$ **then**
        $ngt = gts[keys[d]].length$ // no. of g-truths for d-th $dets$
        $gtObjs = gts[keys[d]].objs$ // g-truths for d-th $dets$
        $det = dets[d].det$ // d-th detection
        for $j = 1$ to ngt **do**
          $ov = IoU(gtObjs[j], det)$ // compute IoU overlap
          if $ov > ovmax$ **then**
            $ovmax = ov$
            $jmax = j$
      if $ovmax \geq iout$ **then**
        if $gtObjs[jmax].assigned == 0$ **then** // if still unassigned
          $tp[d] = 1$ // count as true positive
          $gtObjs[jmax].assigned = 1$ // mark as assigned
        else
          $fp[d] = 1$ // count as false positive
      else
        $fp[d] = 1$
    $fp = cumsum(fp)$ // compute cumulative sum
    $tp = cumsum(tp)$
    // list of recall and precision values at diff. ranks $r = 1, ..., nd$
    $precision = tp/(tp + fp)$; $recall = tp/npos$
    $ap = computeAP(precision, recall)$ // compute area under PR curve
    return ap

Chapter 5

Deep Learning for Detecting Multiple Space-Time Action Tubes in Videos

5.1 Introduction

From the concepts and arguments introduced in Chapter 1 & 2, and from the review of the state of the art conducted in Chapter 3, the following motivating notions emerge:

- deep features are more discriminative and having better generalising capabilities (on large scale real-world image and video data) than shallow features (Section 2) for visual (image & video) representation,
- a single-stage end-to-end trainable deep architecture is a better candidate solution than a multi-stage framework with disjoint optimisation (i.e. a CNN and a set of SVMs are separately trained for action classification) (Section 2.1),
- a CNN trained on specific action detection datasets can generate relatively high quality and cost-effective action region hypotheses than unsupervised region proposal algorithms (Section 2.1),
- a two-stream hypotheses (Section 2.6) is beneficial for action detection and an effective fusion scheme to fuse appearance and motion cues can significantly improve the detection performance,
- posing the inter-frame data association (Section 1.3.4) and temporal localisation (Section 1.3.5) as optimisation problems and solving them using a

dynamic programming approach is a more cost effective and elegant solution than solving them using a graph-based video segmentation and sliding window technique (Section 2.4).

Motivated by these above ideas, in this chapter we propose a novel action detection framework which, instead of adopting an expensive multi-stage pipeline, takes advantage of the most recent single-stage deep learning architectures for object detection [29], in which a fully convolutional neural network (CNN) is trained for generating frame-level region proposals and another CNN-based deep network is trained for both detecting and classifying those proposals in an end-to-end fashion. Subsequently, the appearance and motion based detections (extracted from the respective trained models) are fused using a new test-time fusion technique. Finally, fused frame-level detections are linked in time to form space-time 'action tubes' [14], and then, tubes are temporally trimmed to solve for temporal action localisation. Both temporal linking and localisation problems are solved using two optimisation problems via dynamic programming.

We demonstrate that the proposed action detection pipeline is at least 2× faster in training and 5× faster in test time detection speeds as compared to [14, 20]. In Section 5.4.8, we compare the computing time requirements during training and testing of our approach with [14, 20] on the UCF-101 [17] and J-HMDB-21 [18] datasets. Moreover, our pipeline consistently outperforms [1] previous state-of-the-art results (Section 5.4).

Outline. This chapter is organized as follows. We start by presenting an overview of the approach in Section 5.2. Section 5.3 describes the methodology of the proposed approach. Next, we report the experimental validation of the proposed model in Section 5.4. In Section 5.5, the implementation details are provided. Finally, we present a discussion in Section 5.6.

Related publication. The work presented in this chapter has appeared in BMVC 2016 [33].

5.2 Overview of the approach

Our approach is summarised in Figure 5.1. We train two pairs of Region Proposal Networks (RPN) [29] and Fast R-CNN [49] detection networks - one on RGB

[1] The model presented in this chapter appeared in BMVC 2016 and it outperformed the state-of-the-art results at that time. Later on, the model presented in Chapter 8 has become the state-of-art in the year 2017.

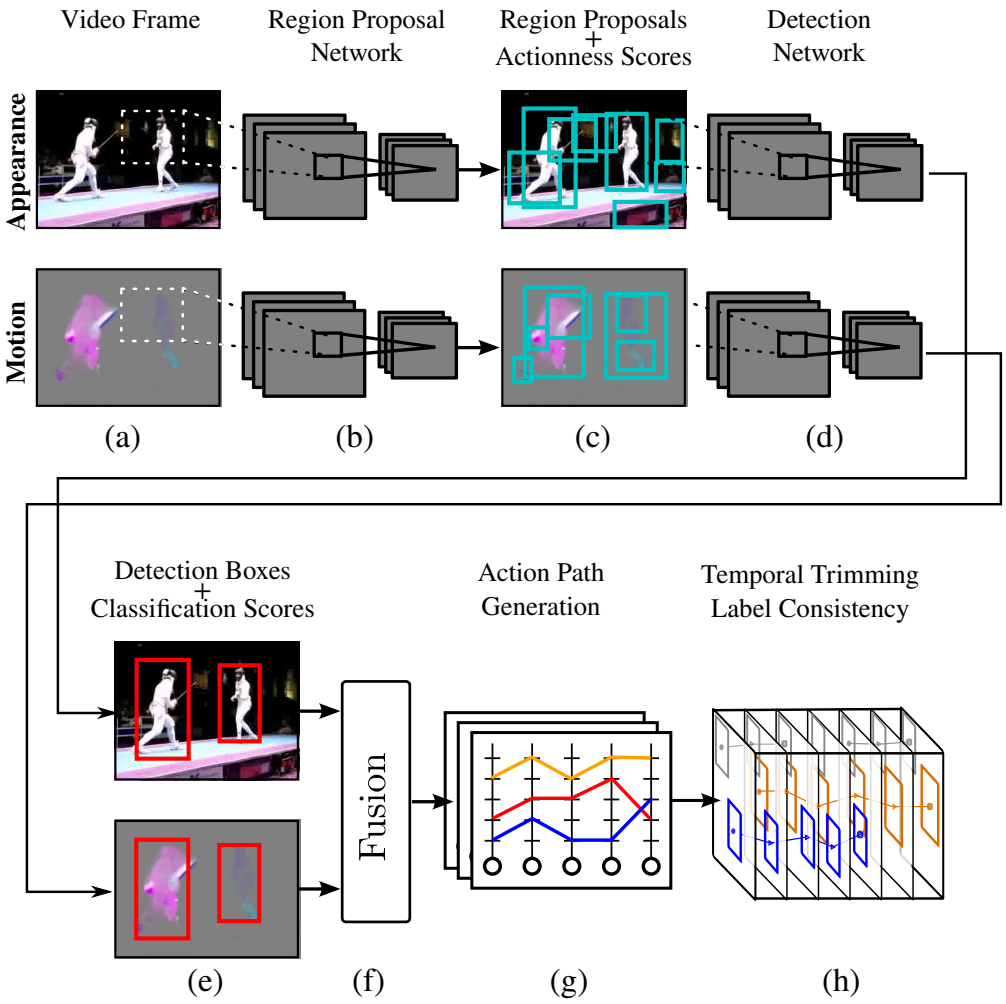


*Figure 5.1: At test time, **(a)** RGB and optical-flow images are passed to **(b)** two separate region proposal networks (RPNs). **(c)** Each network outputs region proposals with associated "actionness" scores (Section 5.3.1). **(d)** Each appearance/motion detection network takes as input the relevant image and RPN-generated region proposals, and **(e)** outputs detection boxes and softmax probability scores (Section 5.3.2). **(f)** Appearance and motion based detections are fused (Section 5.3.3) and **(g)** linked up to generate class-specific action paths spanning the whole video. **(h)** Finally the action paths are temporally trimmed to form action tubes (Section 5.3.4).*

and another on optical-flow images [14]. We refer the readers to Section A.1 for details on optical flow image computation. 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" [2] scores[3]. Next, a Fast R-CNN [29] detection network **(d)** takes as input the original video frame and a subset of the region

[2]The term *actionness* [51] is used to denote the possibility of an action being present within a region proposal.

[3]A softmax score for a region proposal containing an action or not.

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)** (Section 5.3.3). We found 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 [14]). 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 [14]. Furthermore, extracting multiple paths allows our algorithm to account for multiple co-occurring instances of the same action class (see Figure 5.2).

Although it makes use of existing RPN [29] and Fast R-CNN [49] architectures, this work proposes a radically new approach to spatio-temporal 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 [14,20], in which appearance and motion information are fused by combining fc7 (i.e. fully connected layer 7) features, we follow a late fusion approach [27]. 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 5.3.3). The 2nd pass of dynamic programming, we introduce for action tube temporal trimming, contributes to a great extent to significantly improve the detection performance (Section 5.4).

An extensive evaluation on the main action detection datasets demonstrates that our approach significantly outperforms the current state-of-the-art, and is 5 to 10 times faster than the main competitors at detecting actions at test time (Section 5.4). Thanks to our two-pass action tube generation algorithm, in contrast to most existing action classification [27, 54, 96, 97, 131] and localisation [14, 20] approaches, our method is capable of detecting and localising multiple co-occurring action instances in temporally untrimmed videos (see Figure 5.2).

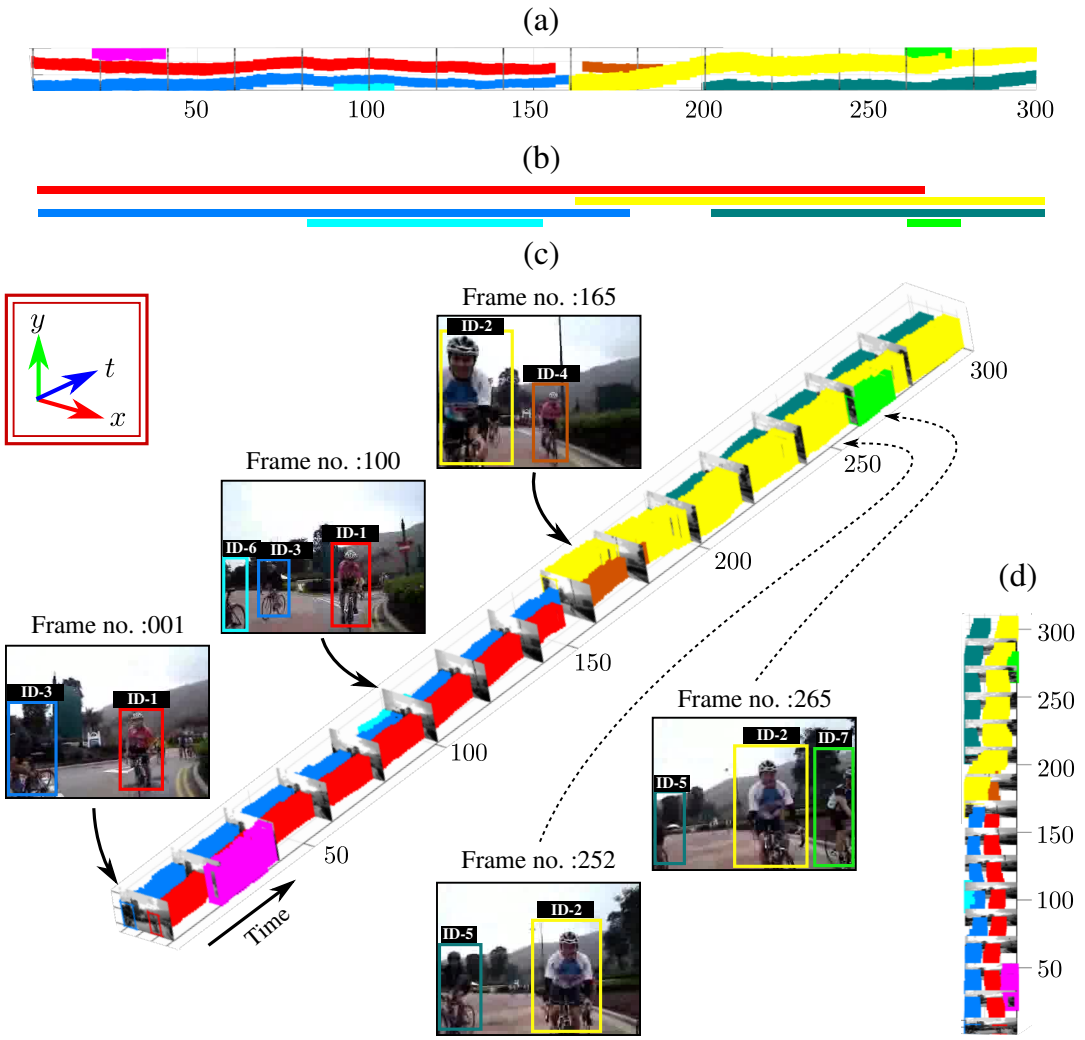


*Figure 5.2: Action tube detection in a 'biking' video taken from UCF-101 [17]. **(a)** Side view of the detected action tubes where each colour represents a particular instance. The detection boxes in each frame are linked up to form space-time action tubes. **(b)** Illustration of the ground truth temporal duration for comparison. **(c)** Viewing the video as a 3D volume with selected image frames; notice that we are able to detect multiple action instances in both space and time. **(d)** Top-down view.*

5.3 Methodology

As outlined in Figure 5.1, our approach combines a region-proposal network (Section 5.3.1-Figure 5.1b) with a detection network (Section 5.3.2-Figure 5.1d), and fuses the outputs (Section 5.3.3-Figure 5.1f) to generate action tubes (Section 5.3.4-Figure 5.1g-h). All components are described in detail below.

5.3.1 Region Proposal Network

To generate rectangular action region hypotheses in a video frame we adopt the Region Proposal Network (RPN) of [29], which is built on top of the last convolutional layer of the VGG-16 architecture by Simonyan and Zisserman [30]. To generate region proposals, this mini-network slides over the convolutional feature map outputted by the last layer, processing at each location an $n \times n$ 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') [29] are generated. We consider those anchors with a high spatial IoU 4.2.4 with the ground truth boxes (IoU > 0.7) as positive examples, whilst those with IoU < 0.3 as negatives. 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. The RPN training objective is explained in Section A.2.

5.3.2 Detection network

For the detection network we use a Fast R-CNN net [49] with a VGG-16 architecture [30]. This takes the RPN-based region proposals (Section 5.3.1) and regresses a new set of bounding boxes for each action class and associates 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.

CNN training strategy. We employ a variation on the training strategy of [29] to train both the RPN and Fast R-CNN networks. Shaoqing *et al.* [29] suggested a 4-steps 'alternating training' algorithm in which in the first 2 steps, a RPN and a Fast R-CNN nets are trained independently, while in the 3$^{\text{rd}}$ and

4^{th} steps the two networks are fine-tuned with shared convolutional layers. In practice, we found empirically that the detection accuracy on UCF101 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^{st} and 2^{nd} steps of [29], while neglecting the 3^{rd} and 4^{th} steps suggested by [29].

5.3.3 Fusion of appearance and motion cues

In a work by Redmon *et al.* [152], the authors combine the outputs from Fast R-CNN and YOLO (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 5.1**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:

$$s_c^*(\mathbf{b}_i^s) = s_c(\mathbf{b}_i^s) + s_c(\mathbf{b}_{max}^f) \times IoU(\mathbf{b}_i^s, \mathbf{b}_{max}^f). \tag{5.1}$$

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 try augmenting the softmax scores of the motion-based detection boxes as per their maximum IoU overlaps with the appearance-based detections, however, it does not give us better results.

An illustration of the above fusion method to combine appearance and motion cues is shown in Figure 5.3.

5.3.4 Action tube generation

The output of our fusion stage (Section 5.3.3) is, for each video frame, a collection of detection boxes for each action category, together with their associated augmented classification scores (Equation 5.1). Detection boxes can then be linked up in time to identify video regions most likely to be associated with

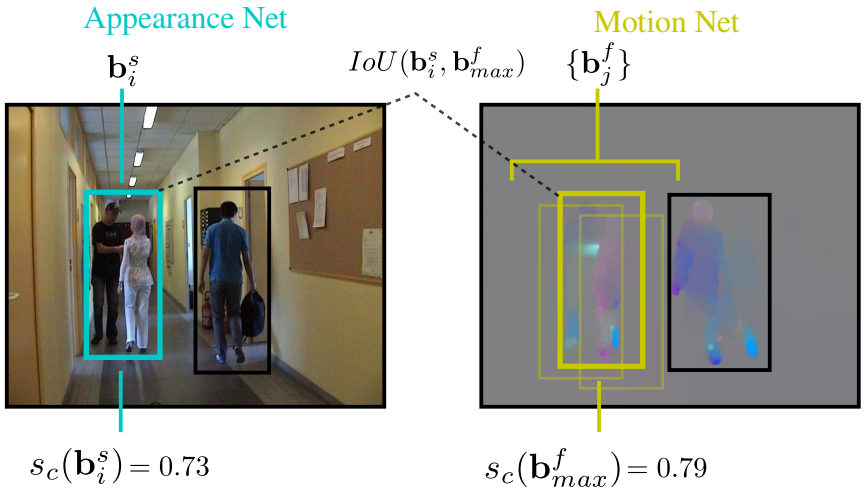


Figure 5.3: For a given test frame and for a specific class c, two sets of detection boxes $\{b_i^s\}$ and $\{b_j^f\}$ are generated by the spatial and flow nets respectively. For a specific box b_i^s, we compute the spatial IoU of b_i^s with $\{b_j^f\}$. If $IoU > \tau$, the softmax score $s_c(b_i^s)$ is augmented using Equation 5.1.

a single action instance, or *action tube*. Action tubes are connected sequences of detection boxes in time, without interruptions, and unlike those in [14] they are not constrained to span the entire video duration. Figure 5.4 illustrates a predicted action tube localising a *"diving"* action in space and time.

They are obtained as solutions to two consecutive energy maximisation problems. First 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 Second, action paths are temporally trimmed by ensuring that the constituting boxes' detection scores are consistent with the foreground label c. Note that, an action path spans the entire video irrespective of the presence of any action. For example, consider a video of duration 100 frames, in which an action is present between frame 25 and 75. However, when we generate the action paths they span the entire video duration, i.e., each path starts at frame 1 and ends at frame 100. For those frames where the action is not present, the action path generation algorithm (Section 5.3.4) picks up suitable detections as per the energy defined for a particular path (Equation 5.2). Subsequently, these noisy detections are expected to be removed from each path during the temporal trimming step (Section 5.3.4).

Building action paths. 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 the a sum of unary and

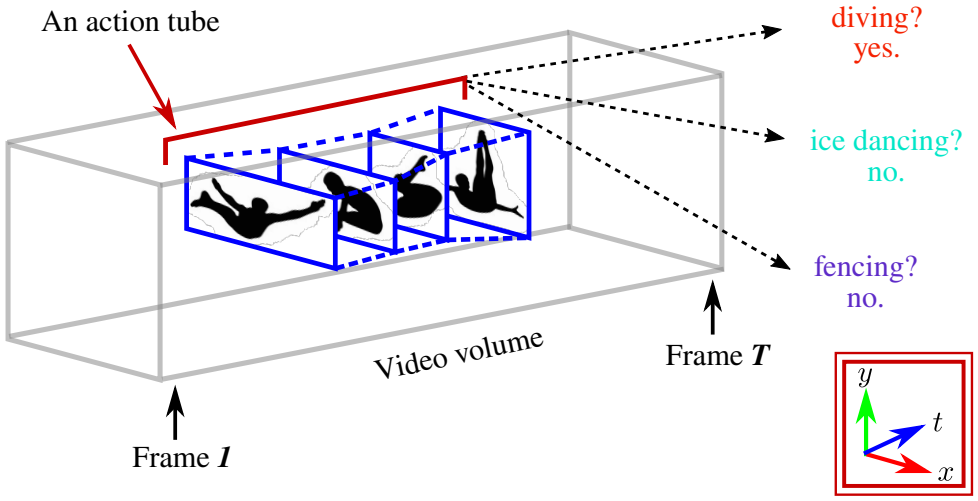


Figure 5.4: Given a "diving" *test video clip, the action detection system is expected to predict an action tube which can localise the* "diving" *action in space and time. The blue rectangular windows denote the frame-level bounding boxes predicted by the system, and the dotted blue lines represent the temporal linking performed by the action tube generation algorithm.*

pairwise potentials:

$$E(\mathbf{p}_c) = \sum_{t=1}^{T} s_c^*(\mathbf{b}_t) + \lambda_o \sum_{t=2}^{T} \psi_o\left(\mathbf{b}_t, \mathbf{b}_{t-1}\right), \tag{5.2}$$

where $s_c^*(\mathbf{b}_t)$ denotes the augmented score (Equation 5.1) 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. The value of the energy (Equation 5.2) 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,

$$\mathbf{p}_c^* = \text{argmax}_{\mathbf{p}_c}\ E(\mathbf{p}_c) \tag{5.3}$$

by simply applying the Viterbi algorithm [14].

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. Figure 5.5 **(a)** illustrates the action path building step in which K class-specific action paths are built by temporally linking frame-level detections $\mathbf{b}_t$ where K is the minimum number of predicted bounding boxes in any frame of the input test video.

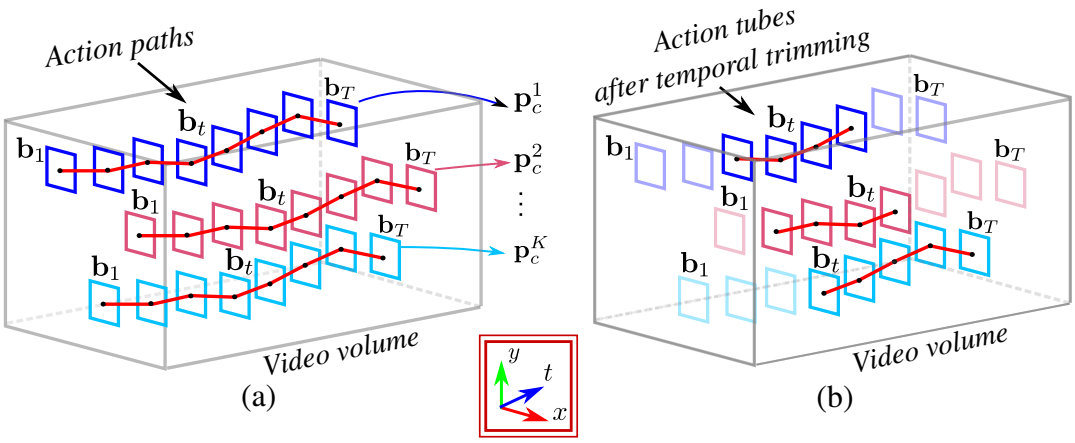


Figure 5.5: ***(a)*** *Class-specific K action paths $\mathbf{p}_c$ are built by temporally linking the frame-level detections $\mathbf{b}_t$.* ***(b)*** *Final action tubes are generated after applying temporal trimming on the K class-specific action paths.*

Smooth path labelling and temporal trimming. As the resulting action-specific paths span the entire video duration, while human actions typically only occupy a fraction of it, temporal trimming becomes necessary. The first pass of dynamic programming (Equation 5.2) 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 5.1); and ii) is smooth (no sudden jumps).

As in the previous pass, we may solve for the best labelling by maximising:

$$\mathbf{L}^*_{\mathbf{p}_c} = \underset{\mathbf{L}_{\mathbf{p_c}}}{\operatorname{argmax}} \left(\sum_{t=1}^{T} s_{l_t}(\mathbf{b}_t) - \lambda_l \sum_{t=2}^{T} \psi_l\left(l_t, l_{t-1}\right) \right), \tag{5.4}$$

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

$$\psi_l(l_t, l_{t-1}) = \begin{cases} 0 & \text{if } l_t = l_{t-1} \\ \alpha_c & \text{otherwise,} \end{cases} \tag{5.5}$$

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 detec-

tion accuracy (Section 5.4.6). Equation 5.5 is the standard Potts model which penalises labellings that are not smooth, thus enforcing a piecewise constant solution. Again, we solve (Equation 5.4) using the Viterbi algorithm.

All contiguous subsequences 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 5.1) of its constituting detection boxes. Note that, an action tube can contain only a single action instance. A single action instance represents the spatial and temporal locations of an action in a video. The spatial location is defined by a bounding box at each frame (where the action is present), and the temporal location is the length of time for which the action happens, i.e., the lifespan or duration of that action. If an action occurs multiple times in a video, then each occurrence of that action represents a different action instance and are detected by different action tubes. Figure 5.5 **(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. For more details on the action tube problem formulation, please refer to Section A.5 .

5.4 Experimental validation

We use the following abbreviations to denote three different action detection networks: (1) apnet - appearance detection network, (2) monet - motion detection network and (3) amnet - appearance- and motion-based fused model.

5.4.1 Performance comparison on UCF-101

Table 5.1 presents the results we obtained on UCF-101, and compares them to the previous state-of-the-art [20, 26]. We achieve an mAP of 66.36% compared to 46.77% reported by [20] (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% [20] at $\delta = 0.2$). Note that we are the first to report results on UCF-101 up to $\delta = .6$, attesting to the robustness of our approach to more accurate localisation requirements. Although our separate appearance- and motion-based detection pipelines already outperform the state-of-the-art (Table 5.1), their combination (Section 5.3.3) delivers a significant performance

Table 5.1: Quantitative action detection results (mAP) on the UCF-101 dataset.

Spatio-temporal overlap threshold δ	0.05	0.1	0.2	0.3	0.4	0.5	0.6
FAP [26]	42.80	–	–	–	–	–	–
STMH [20]	54.28	51.68	46.77	37.82	–	–	–
Our (apnet)	67.56	65.45	56.55	48.52	39.00	30.64	22.89
Our (monet)	65.19	62.94	55.68	46.32	37.55	27.84	18.75
Our (amnet)	**78.85**	**76.12**	**66.36**	**54.93**	**45.24**	**34.82**	**25.86**

apnet - appearance network, monet - motion network, amnet - fused model.

increase.

Figure 5.6: Action detection/localisation results on UCF101. 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.

Some representative example results from UCF-101 are shown in Figure 5.6. Our method can detect several (more than 2) action instances concurrently, as shown in Figure 5.2, in which three concurrent instances and in total six action instances are detected correctly. Quantitatively, we report class-specific video AP (average precision in %) of 88.0, 83.0 and 62.5 on the UCF-101 action categories *'fencing'*, *'salsa spin'* and *'ice dancing'*, respectively, which all concern multiple inherently co-occurring action instances. Class-specific video APs on UCF-101

are reported in Section 5.4.5.

5.4.2 Performance comparison on J-HMDB-21

The results we obtained on J-HMDB-21 are presented in Table 5.2. Our method again outperforms the state-of-the-art, with an mAP increase of 18% and 11% at $\delta = .5$ as compared to [14] and [20], 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 $\delta = .5$. These results attest to 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 5.4.4). Sample detections on J-HMDB-21 are shown in Figure 5.7. Also, we list our classification accuracy results on J-HMDB-21 in Table 5.3, where it can be seen that our method achieves an 8% gain compared to [14].

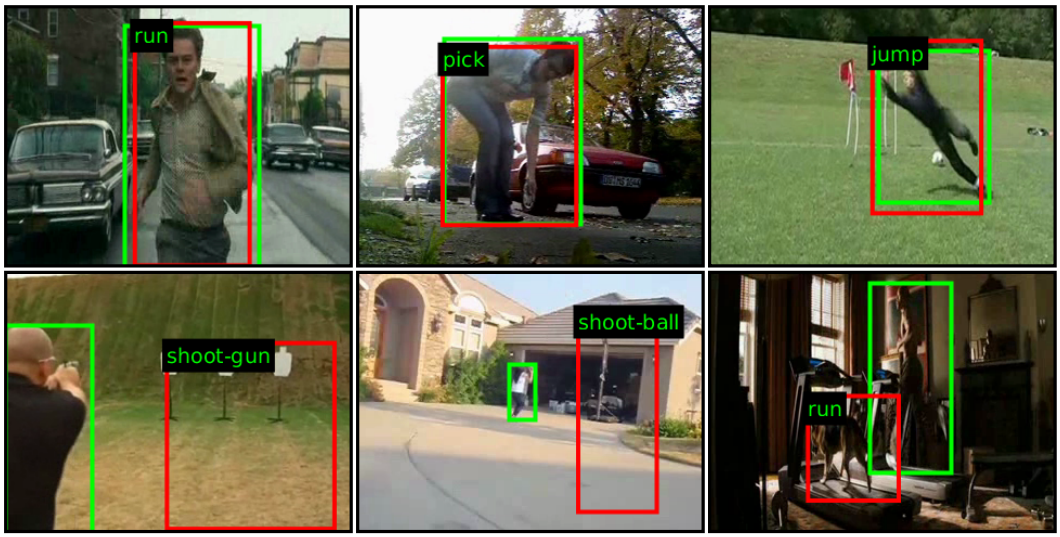


Figure 5.7: Sample space-time action localisation results on JHMDB. Top row: accurate detection examples. Bottom row: mis-detection examples.

5.4.3 Performance comparison on LIRIS-HARL

LIRIS HARL allows us to demonstrate the efficacy of our approach on temporally untrimmed videos with co-occurring actions. For this purpose we use LIRIS-HARL's specific evaluation tool - the quantitative results are shown in Table 5.4 and 5.5 and the qualitative results can be visualised in Figure 5.8. Our

Table 5.2: Quantitative action detection results (mAP) on the J-HMDB-21 dataset.

Spatio-temporal overlap threshold δ	0.1	0.2	0.3	0.4	0.5	0.6	0.7
ActionTube [14]	–	–	–	–	53.3	–	–
Wang *et al.* [134]	–	–	–	–	56.4	–	–
STMH [20]	–	63.1	63.5	62.2	60.7	–	–
Our (apnet)	52.99	52.94	52.57	52.22	51.34	49.55	45.65
Our (monet)	69.63	69.59	69.49	69.00	67.90	65.25	54.35
Our (amnet)	**72.65**	**72.63**	**72.59**	**72.24**	**71.50**	**68.73**	**56.57**

apnet - appearance network, monet - motion network, amnet - fused model.

Table 5.3: Classification accuracy on the J-HMDB-21 dataset.

Method	Wang *et al.* [54]	STMH [20]	ActionTube [14]	Our (appearance+motion fusion)
Accuracy (%)	56.6	61	62.5	**70.0**

results are compared with those of i) VPULABUAM-13 [143] and ii) IACAS-51 [144] from the original LIRIS HARL detection challenge. In this case, our method outperforms the competitors by an even larger margin. We report space-time detection results by fixing the threshold quality level to 10% for the four thresholds [151] and measuring temporal precision and recall along with spatial precision and recall, to produce an integrated score. We refer to Section 4.2.5 for more details on LIRIS HARL evaluation metrics. Note that, ours is the first published work which reports action detection results on LIRIS HARL dataset. It would be of great value to evaluate the action detection performance of other competitive methods [14,20] on LIRIS HARL. However, [14]'s approach can solve only spatial action localisation in temporally trimmed videos, and thus, not suitable for LIRIS HARL which has all temporally untrimmed videos. Besides, [20]'s approach uses a highly expensive detection pipeline (Section 2.1). As a future work we plan to evaluate [20]'s approach on LIRIS HARL dataset.

We also report in Table 5.6 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 $\delta = .2$ due to our fusion strategy. To demonstrate the advantage of our 2nd pass of DP (Section 5.3.4), we also generate results (mAP) using only the first DP pass (Section 5.3.4). Without the 2^{nd} pass performance decreases by 20%, highlighting the importance of temporal trimming in the construction of action tubes.

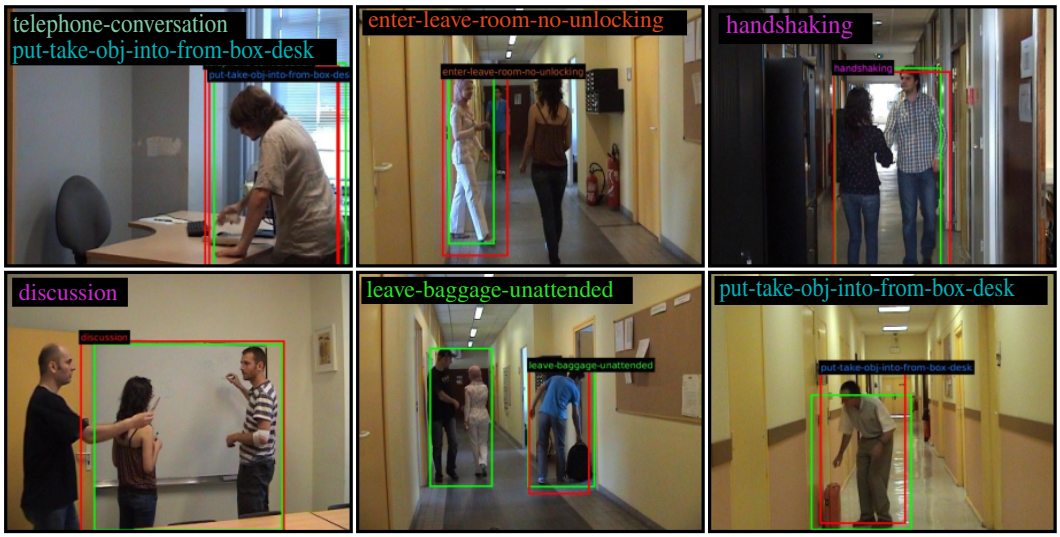


Figure 5.8: Frames from the space-time action detection results on LIRIS-HARL, some of which include single actions involving more than one person like 'handshaking' and 'discussion'. Top row: accurate detection examples. Bottom row: mis-detection examples.

Table 5.4: Quantitative action detection results on the LIRIS-HARL dataset.

Method	Recall-10	Precision-10	F1-Score-10
VPULABUAM-13-IQ [143]	0.04	0.08	0.05
IACAS-51-IQ [144]	0.03	0.04	0.03
(**Ours**)	**0.568**	**0.595**	**0.581**

Table 5.5: Quantitative action detection results on the LIRIS-HARL dataset.

Method	I_{sr}	I_{sp}	I_{tr}	I_{tp}	IQ
VPULABUAM-13-IQ [143]	0.02	0.03	0.03	0.03	0.03
IACAS-51-IQ [144]	0.01	0.01	0.03	00.0	0.02
(**Ours**)	**0.5383**	**0.3402**	**0.4802**	**0.4739**	**0.458**

I_{sr} - integrated spatial recall, I_{sp} - integrated spatial precision,
I_{tr} - integrated temporal recall, I_{tp} - integrated temporal precision,
IQ - integrated quality or performance measure [151].

Table 5.6: Quantitative action detection results (mAP) on LIRIS-HARL for different δ.

Spatio-temporal overlap threshold δ	0.1	0.2	0.3	0.4	0.5
Appearance detection model	46.21	41.94	31.38	25.22	20.43
Motion detection model	52.76	46.58	35.54	26.11	19.28
Appearance+motion fusion with one DP pass	38.1	29.46	23.58	14.54	9.59
Appearance+motion fusion with two DP passes	**54.18**	**49.10**	**35.91**	**28.03**	**21.36**

5.4.4 Comparative analysis of region proposal quality

We analyse the quality of Selective Search vs RPN-based region proposals using the Recall-to-IoU measure [29]. First, We extract Selective Search (SS) boxes (approximately 1000 boxes/frame) and RPN-based detection boxes (300 boxes/frame) from our detection network on UCF-101 testsplit-1. Also, we apply a constraint on the RPN-based proposals by putting a threshold to their class-specific softmax probability scores s_c and only considering those proposals with $s_c \geq 0.2$. For each UCF-101 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 5.9.

5.4.5 Ablation study

We are the first to report an ablation study of the spatio-temporal action localisation performance on UCF-101 dataset. Table 5.7 shows the class-specific video AP (average precision in %) for each action category of UCF-101 generated by the appearance- and motion-based detection networks separately, and by the *appearance+motion* fusion model. Results are generated at a spatio-temporal overlap threshold of $\delta = 0.2$. For 18 out of 24 action classes, our *appearance+motion* fusion technique gives the best APs. The appearance-based detection net alone achieves the best APs for two classes: *horse riding*(HR) and *tennis swing*(TS), while the motion-based detection net outperforms for action classes: *cricket bowling*(CB), *long jump*(LJ), *salsa spin*(SaS) and *soccer juggling* (SJ). It is worth noting that for action classes HR and TS, static appearance cues such as "horse" and "tennis player" are the most discriminative features whereas, for action classes CB, LJ, SaS and SJ, the motion's temporal dynamics seems to be most discriminative. This could explain the highest APs of appearance- and motion-based networks for these specific actions.

5.4.6 Impact of label smoothing on detection performance

We also conduct experiments to show the significance of the path label smoothing step. More specifically, 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 spatio-temporal detection accuracy. First, we generated detection results on UCF-101 test set (split-1) by setting the constant parameter

Table 5.7: An ablation study of the spatio-temporal detection results (video APs in %) on UCF-101.

Actions	Basketball	BasketballDunk	Biking	CliffDiving	CricketBowling	Diving
appearance	30.5	22.7	56.1	44.2	11.5	89.7
motion	22.9	41.5	52.0	64.6	**30.2**	86.7
appearance+motion	**36.7**	**48.3**	**60.4**	**73.2**	19.9	**96.6**
Actions	GolfSwing	HorseRiding	IceDancing	LongJump	PoleVault	RopeClimbing
appearance	59.9	**95.4**	59.2	41.5	48.9	77.8
motion	47.0	91.5	62.0	**68.3**	51.9	88.2
appearance+motion	**66.5**	94.1	**62.5**	55.7	**72.6**	**89.6**
Actions	Skiing	Skijet	SoccerJuggling	Surfing	TennisSwing	TrampolineJumping
appearance	68.4	88.0	34.6	55.7	**34.3**	50.3
motion	51.8	61.6	**87.6**	42.7	08.6	31.1
appearance+motion	**78.9**	**92.8**	86.4	**61.3**	32.6	**51.3**
Actions	Fencing	FloorGymnastics	SalsaSpin	SkateBoarding	VolleyballSpiking	WalkingWithDog
appearance	86.9	93.8	52.4	76.5	13.2	73.3
motion	83.4	80.0	**83.0**	67.0	07.3	63.8
appearance+motion	**88.0**	**99.7**	57.5	**85.0**	**15.9**	**75.6**

Table 5.8: Spatio-temporal detection results (mAP) on UCF-101 using two different sets of α_c values.

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

$\alpha_c = 0$ for each action category. Then, we use the cross validated class-specific α_c values and again generated detection results. In our experiment, we set the spatio-temporal IoU threshold to $\delta = 0.2$. Table 5.8 presents the results for the following two cases: **(a)** detection result obtained by setting $\alpha_c = 0$ for all action class, and **(b)** detection result generated using the cross validated class-specific α_c values. Table 5.9 shows the class-specific α_c values obtained by cross validation. Notice, the class-specific α_c improves the detection accuracy (mAP) by 6%. We empirically observe that the class-specific softmax probability scores (from detection network) are not always stable throughout an action path generated by the $1st$ pass of DP algorithm, i.e., there are sudden jumps in the scores causing a valid action path to be broken by the $2nd$ pass DP algorithm. The class-specific α_c value helps to stabilise an action path by introducing a certain penalty in the 2nd pass of DP. Due to the fact that each action category has its own temporal duration and speed, having different alpha values for different action classes is better than having a single alpha value assigned for all classes. One interesting point to note here that, there might exist a single alpha ($\alpha > 0$) value for all classes which can give us the best mAP. In near future, we would like to investigate on it.

Table 5.9: Class specific α_c values for each action category in UCF-101 obtained from cross validation.

Actions	Basketball	BasketballDunk	Biking	CliffDiving	CricketBowling	Diving
class-specific α_c	0	0.8	0	14	0	0.2
Actions	GolfSwing	HorseRiding	IceDancing	LongJump	PoleVault	RopeClimbing
class-specific α_c	0.2	4	18	0	1	0.8
Actions	Skiing	Skijet	SoccerJuggling	Surfing	TennisSwing	TrampolineJumping
class-specific α_c	2	10	0.2	0	0.2	0
Actions	Fencing	FloorGymnastics	SalsaSpin	SkateBoarding	VolleyballSpiking	WalkingWithDog
class-specific α_c	0	0.6	6	8	2	0.2

5.4.7 Additional qualitative detection results

Figure 5.10 provides additional evidence on the temporal detection and spatial localisation performance of our method.

5.4.8 Computing time analysis for training and testing

We compare detection speed at test time of the combined region proposal generation and CNN feature extraction approach used in ([14, 20]) to our neural-net based, single stage action proposal and classification pipeline on the J-HMDB-21 dataset.We find our method to be 10× faster than [14] and 5× faster than [20], with a mean of 113.52 [14], 52.23 [20] and 10.89 (ours) seconds per video, averaged over all the videos in J-HMDB-21 split1.

We also analyse training and testing time requirements of our method in comparison with our main competitors [14, 20]. Note that [20] modifies the pipeline of ActionTube [14] by adding a 'tracking by detection' module - thus in Table 5.10 and 5.11, while comparing the computation time, we only consider those components of the detection pipelines which are common to both [14] and [20].

Comparison on UCF-101 dataset. The first comparison is run on the UCF-101 dataset, using 7 NVIDIA Titan X GPUs. Time is computed assuming that appearance- and motion-based CNNs are trained in parallel. Our method is at least 2× faster in training and 20× faster in testing on UCF101 (refer Table 5.10). The most time consuming step in [14, 20] is CNN feature extraction, as CNN features are extracted for each region proposal and for each video frame, and each feature extraction process requires to run a CNN forward pass. For example, using ActionTube [14]'s approach, for UCF-101's $240k$ training video frames with

Table 5.10: Training and test time detection speed comparison on UCF-101 with [14, 20].

Training time: time computed on 2293 UCF-101 training video clips (split-1)			
ActionTube [14] and STMH [20]		Ours	
Fine-tuning CNNs	12 hours	RPN training	1 day
CNN feature extraction	5 days	Region proposal extraction	26 minutes
One vs rest SVMs training	1 day	Fast R-CNN training	2 days
Total training time required	6+ days	**Total training time required**	**3+ days**
Test time: time computed on 914 UCF-101 test video clips (split-1)			
CNN feature extraction	2 days	Region proposal extraction	20 minutes
		Fast R-CNN detections	38 minutes
		1^{st} pass DP	76 minutes
		2^{nd} pass DP	7 minutes
Total test time required	2+ days	**Total test time required**	**2.5 hours**

approximately 100 Selective Search based region proposals per frame, we need $240k \times 100$ CNN forward passes to extract features there. In contrast an RPN net needs only $240k$ CNN forward passes, as it uses a single shared convolutional feature map for proposal generation and requires only one CNN forward pass per video frame.

Even in our pipeline RPN region proposal extraction is time consuming. A RPN model takes $100ms$ to process each frame - multiplied by $240k$ UCF-101 training video frames, the entire process takes 7 hours. We significantly reduce this time to $\sim$ 26 minutes by employing 7 NVIDIA Titan X GPUs in parallel to extract region proposals. Time computation for the competing methods is reported considering $40k$ training iterations for CNN fine-tuning; for RPN and Fast R-CNN training $320k$ CNN training iterations are used. Testing time performances for the proposed method are once again reported while using 7 Titan X GPUs in parallel.

Test-time detection speed comparison on J-HMDB-21. We compare the video-level detection time of our proposed pipeline with the state-of-the-art [14, 20] which use an expensive multi-stage classification strategy. We report comparison results on J-HMDB-21 dataset. We exclude our 2^{nd} pass DP step due to the fact that J-HMDB-21 video clips do not require temporal trimming. Our 1st pass DP and optical flow based 'motion frame' generation steps are common to [14] and our pipeline, and thus, we 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 [14, 20] with our RPN

Table 5.11: Test time detection speed comparison on J-HMDB-21 with [14, 20].

ActionTube [14], STMH [20]	Average time (Sec./video)
Selective Search [14] / EdgeBoxes [20]	68.10 / 6.81
CNN feature extraction	45.42
Avg. detection time	113.52 [14] / 52.23 [20]

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

and detection nets computation times. Table 5.11 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. All the Experimental results were generated using a desktop computer with an Intel Xeon CPU@3.20GHz and NVIDIA Titan X GPU. Our method is at least 10× faster than [14] and 5× than [20] for those steps we compared with (see Table 5.11).

5.5 Implementation details

5.5.1 Generating correct ground truth annotations

We downloaded the original XML annotation files available online at: `http://crcv.ucf.edu/data/UCF101.php` and parsed them using MATLAB XML parser. We found a mismatch between the number of actual ground truth action tubes present in the XML files and the number of tubes available in the annotations of [20, 24]. [20, 24]'s annotations did not contain all the ground truth action tubes. Subsequently, we developed a MATLAB based parser which takes the annotation XML files as input and generates a MATLAB structure which contains all the valid ground truth action tubes both for train and test sets. All the results on UCF-101-24 dataset in this dissertation were generated using our new corrected ground truth annotation which is available online at `https://bitbucket.org/sahasuman/bmvc2016_code`. Following to our corrections, [24, 32] corrected their action detection results on UCF-101-24 dataset.

5.5.2 Modifications in the existing codebase

We downloaded the publicly available Faster R-CNN MATLAB code from `https://github.com/ShaoqingRen/faster_rcnn` to train the RPN and Fast R-CNN networks. We practically experienced a shortage of RAM memory while training UCF-101 using this code. The original MATLAB code tries to load the entire training data into RAM. For datasets such as UCF-101 the amount of training data is substantial, causing out-of-memory issues. For example, in UCF-101, we have $240k$ training video frames, a horizontal flipping process for each video frame gives us in total $480k$ training frames. Loading RPN training data for $480k$ frames takes more than $64GB$ of RAM in our experiments. The situation becomes worse in Fast R-CNN training, when the code tries to load training data for $480k \times 2000$ region proposals which exhausts the entire $128GB$ of RAM completely. In the default setting, a RPN net takes as input 1 video frame per training iteration and a Fast R-CNN takes as input 2 frames per iteration. Thus, loading the entire training data into RAM can be easily avoided by caching the frame-level training data into disk storage and fetching them as and when required by the CNN training module. We modified the existing MATLAB code to require a smaller amount of RAM memory for both RPN and Fast R-CNN training.

5.5.3 Selective Search region proposals

In Section 5.4.4 and 5.4.8, we presented a comparative analysis of region proposal quality and train and test time detection speed comparison with the state-of-the-art. In these experiments, we used the Selective Search algorithm to extract region proposals on UCF-101 video frames. We extracted Selective Search (SS) region proposals using the publicly available code from `https://github.com/rbgirshick/rcnn`. We used the SS's 'fast mode' and obtained approximately 1000 SS boxes per video frame, subsequently, we filtered these SS boxes using the motion saliency scores of [14] and retain on average 100 SS boxes per frame.

5.5.4 Network training and testing

Training data preparation We divided the UCF-101 train split one into two subsets. The first subset consists of 70% (1605 videos $\sim 240k$ frames) and the second subset contains 30% (688 videos) of the training videos from UCF-101 train split one. 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 J-HMDB-21 and LIRIS HARL D2 datasets, we used the original training sets provided by the authors [18, 151].

CNN weight initialisation The RPN and Fast R-CNN networks [29] were initialised with weights from a pre-trained ImageNet model [153].

CNN solver configuration setting. For UCF-101, 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 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 takes 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.

5.6 Discussion

The superior performance of the proposed method is due to a number of reasons. 1) Instead of using unsupervised region proposal algorithms as in [47, 50], our pipeline takes advantage of a supervised RPN-based region proposal approach which exhibits better recall values than [47] (see Section 5.4.4). 2) Our fusion technique improves the mAPs (over the individual appearance or motion models) by 9.4%, 3.6% and 2.5% on the UCF-101, J-HMDB-21 and LIRIS HARL datasets respectively. We are the first to report an ablation study (Section 5.4.5) where it is shown that the proposed fusion strategy (Section 5.3.3) improves the class-specific video APs of UCF-101 action classes. 3) Our original 2nd pass of DP is responsible for significant improvements in mAP by 20% on LIRIS HARL (Section 5.4.3) and 6% on UCF-101 (Section 5.4.6). Additional qualitative results are provided in the supplementary video [4], and on the project web page[5], where the code has also been made available.

[4] `https://www.youtube.com/embed/vBZsTgjhWaQ`
[5] `http://sahasuman.bitbucket.io/bmvc2016`

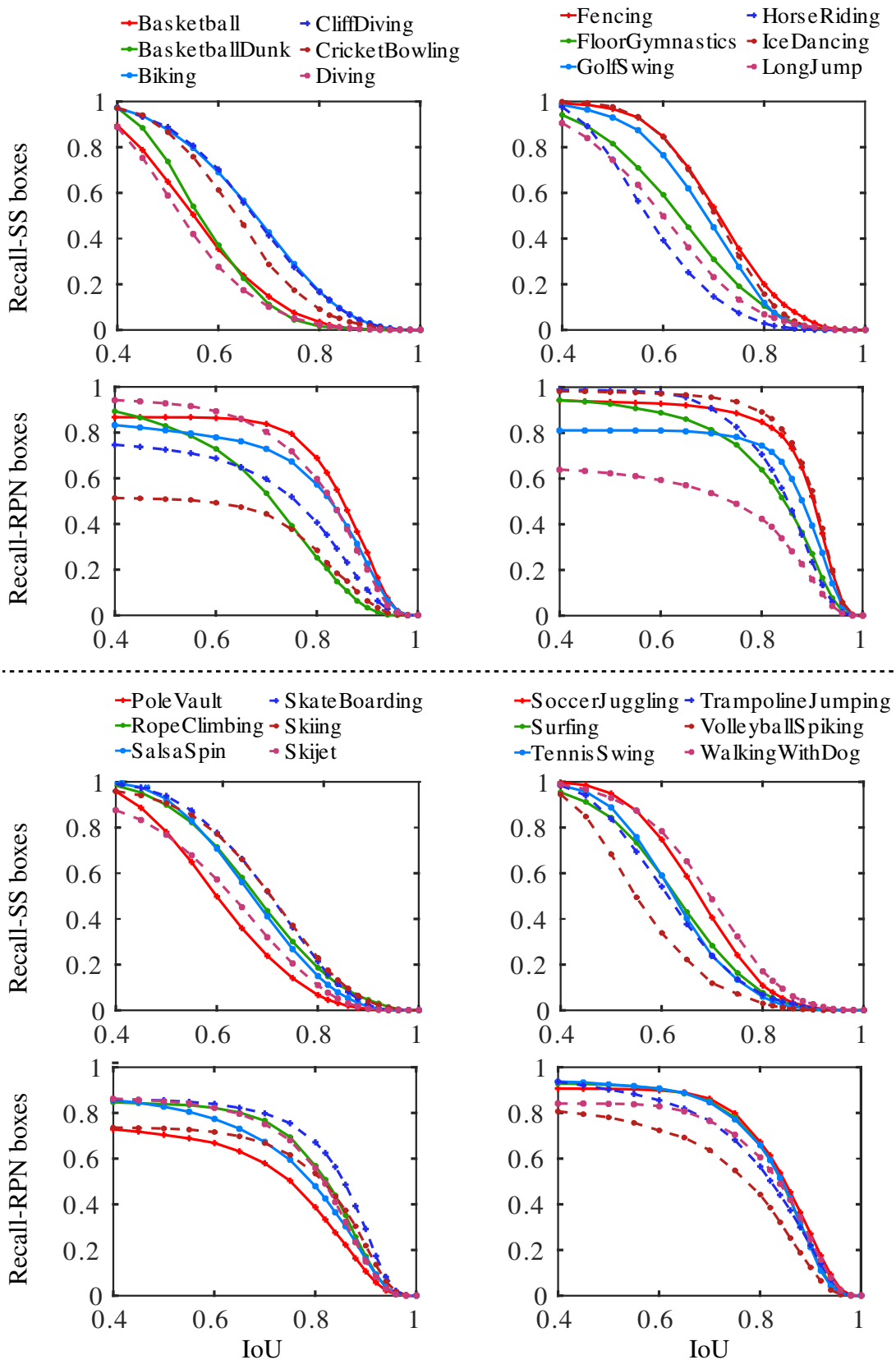


Figure 5.9: Performance comparison between Selective Search (SS) and RPN-based region proposals on 24 action classes of UCF-101-24 dataset. 1^{st} & 3^{rd} rows: recall vs. IoU curve for Selective Search based region proposals; 2^{nd} & 4^{th} rows: recall vs. IoU curve for RPN-based region proposals.

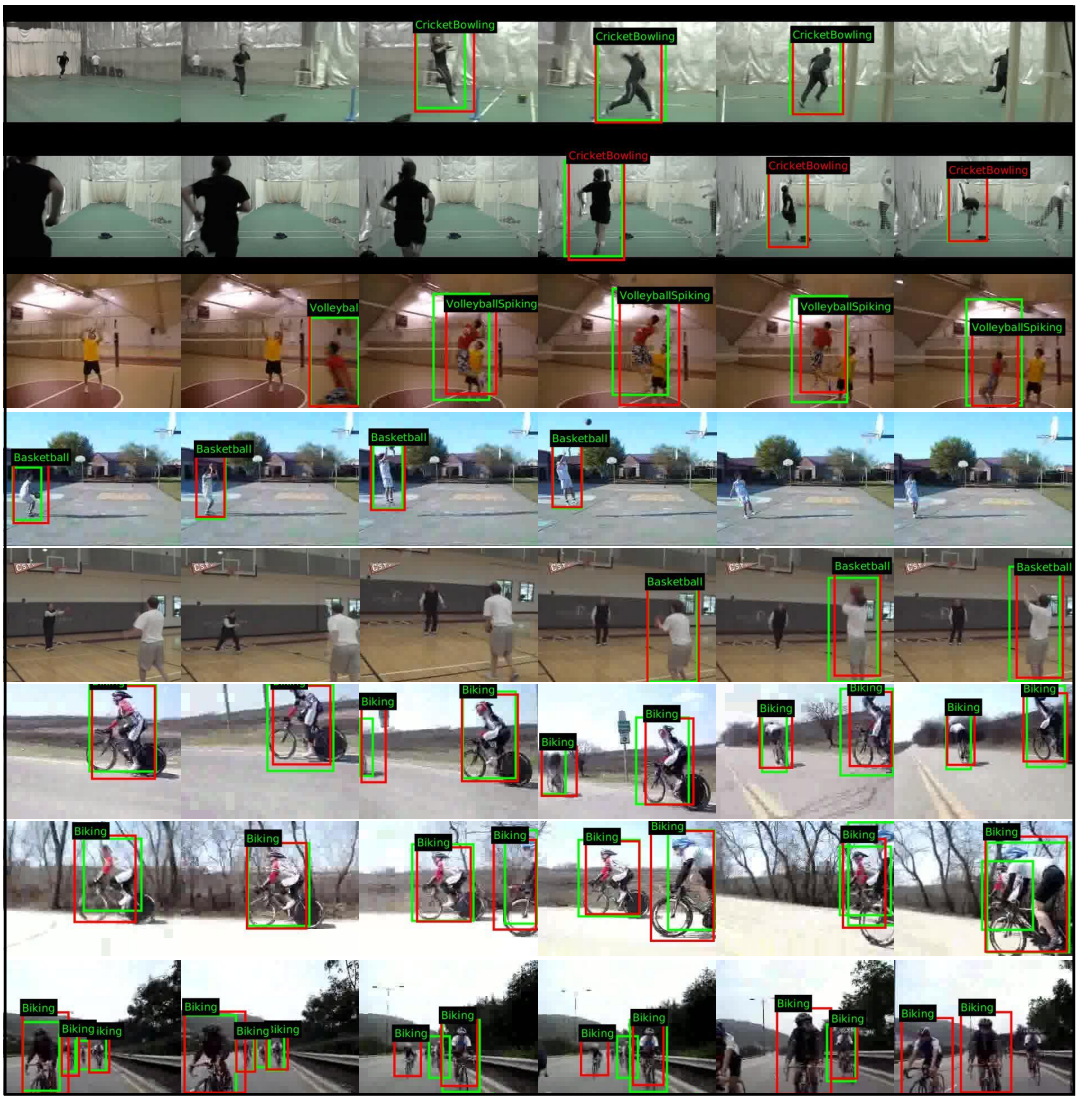


Figure 5.10: Sample qualitative spatio-temporal localisation results on UCF-101. Each row represents a UCF-101 test video clip. Ground-truth bounding boxes are in green, detection boxes in red.

Chapter 6

Spatio-temporal Action Instance Segmentation and Localisation

6.1 Introduction

The action detection framework presented in Chapter 5 along with other competing approaches [14, 20, 32, 33] address the problem of action detection in a setting where videos contain single action category and most of them are temporally trimmed. In contrast, this chapter addresses the problems of both spatio-temporal action instance segmentation (Section 2.5) and action detection. Here, we consider real-world scenarios where videos often contain co-occurring action instances belong to different action categories. Consider the example shown in Figure 6.1, where our proposed model performs action instance segmentation and detection of two co-occurring actions *"leaving bag unattended"* and *"handshaking"* which have different spatial and temporal extents within the given video sequence. The video is taken from the LIRIS-HARL dataset [18]. In this chapter, we propose a deep learning based framework for both action instance segmentation and detection, and evaluate the proposed model on the LIRIS-HARL dataset which is more challenging than the standard benchmarks: UCF-101-24 [17] and J-HMDB-21 [18] due to its multi-label and highly temporally untrimmed videos (Chapter 4). To demonstrate the generality of the segmentation results on other standard benchmarks, we present some additional qualitative action instance segmentation results on the standard UCF-101-24 dataset (Section 6.4.4).

Outline. This chapter is organized as follows. First we present an overview of the approach in Section 6.2. We then introduce the detailed methodology in Section 6.3. Finally, Section 6.4 and Section 6.5 present the experimental

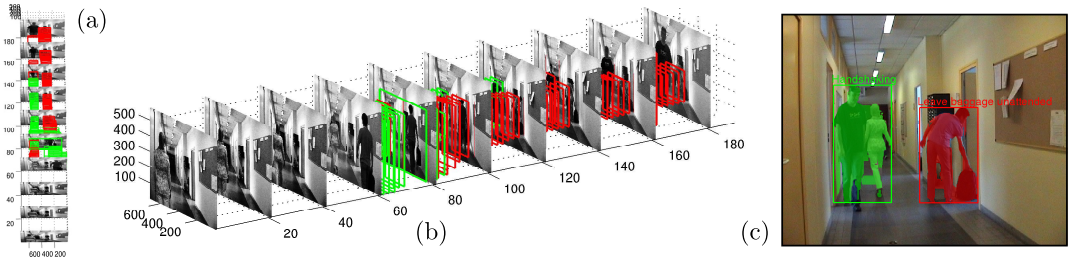


Figure 6.1: A video sequence taken from the LIRIS-HARL dataset plotted in space-and time. ***(a)*** *A top down view of the video plotted with the detected action tubes of class* "handshaking" *in green, and* "person leaves baggage unattended" *in red. Each action is located to be within a space-time tube.* ***(b)*** *A side view of the same space-time detections. Note that no action is detected at the beginning of the video when there is human motion present in the video.* ***(c)*** *Action instance segmentation results for two actions occurring simultaneously in a single frame.*

validation and discussion respectively.

Related publication. The work presented in this chapter has appeared in arXiv [37].

6.2 Overview of the approach

An overview of the algorithm is depicted in Figure 6.2. At test time, we start by performing binary human motion segmentation **(a)** for each input video frame by leveraging the human action segmentation [31], followed by a frame-level region proposal generation**(b)** (Section 6.3.1). Proposal bounding boxes are then used to crop patches from both RGB and optical flow frames **(c)**. We refer readers to Section A.1 for details on optical flow frame computation. Crop image patches are resized to a fixed dimension and fed as inputs to an appearance- and a motion-based detection network **(d)** (Section 6.3.2) to compute CNN fc7 features. Subsequently, these appearance- and motion-based fc7 features are fused, and later, these fused features are classified by a set of one-vs-all SVMs. Each fused feature vector is a high-level image representation of its corresponding warped region and encodes both static appearance (e.g. boundaries, corners, object shapes) and motion pattern of human actions (if there is any). Finally, the top k frame-level detections (regions with high classification scores) are temporally linked in time (Section 5.3.4) to build class-specific action tubes **(e)** and then, these tubes are trimmed (Section 5.3.4) to solve for temporal action localisation (Section 6.3.5). Pixels belonging to each action tube are assigned class- and instance-aware action labels by taking advantage of both tube's class

score and the binary action segmentation maps computed in **(a)**. At train time, first action region hypotheses are generated for RGB video frames using Selective Search [47] (Section 6.3.1), then, pretrained appearance and motion CNNs **(d)** are fine-tuned on the warped regions extracted from both RGB and flow frames. Subsequently, fine-tuned appearance and motion CNNs are used to compute fc7 features from both RGB and flow training frames, features are then fused and pass as inputs to a set of one-vs-all SVMs for training. A detailed descriptions of these above steps are presented in Section 6.3. Note that, here we follow a R-CNN based [28] multi-stage training strategy (Section 2.1) and feature fusion scheme which are different from the single-stage training (of Faster R-CNN [29]) and fusion technique used in Chapter 5.

6.3 Methodology

6.3.1 Region proposal generation

We denote each 2D region proposal '**r**' as a subset of the image pixels, associated with a minimum bounding box '**b**' around it. In the following sub sections we present our two different region proposal generation schemes: (1) the first one is based on human motion segmentation algorithm [31], and (2) the second one uses Selective Search algorithm [47] to generate 2D action proposals.

Proposals based on motion segmentation. The human motion segmentation [31] algorithm generates binary segmentation of human actions (Figure 6.2 **(a)**). It extracts human motion from video using long term trajectories [39]. In order to detect static human body parts which don't carry any motion but are still significant in the context of the whole action, it attaches scores to these regions using a human shape prior from a deformable part-based (DPM) model [91]. By striking balance between the human motion and static human-body appearance information, it generates binary silhouettes of human actions in space and time. At test time our region proposal algorithm accepts the binary segmented images produced by [31], and generates region proposal hypotheses using all possible combinations of 2D connected components ($2^N - 1$) present in the binary map (Figure 6.2 **(b)**), where N is the number of 2D connected components present in each video frame (Section A.3). In the following subsection, we briefly introduce the human motion segmentation pipeline.

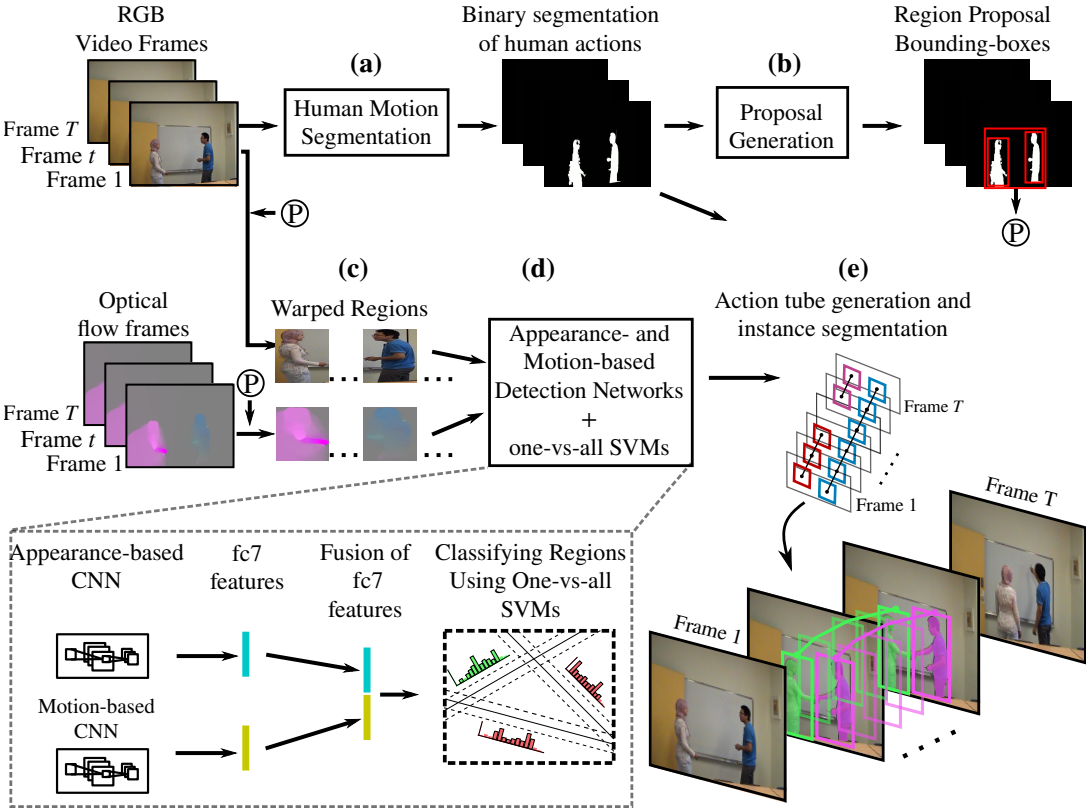


Figure 6.2: Overview of the proposed spatio-temporal action instance segmentation and detection pipeline. At test time, ***(a)*** *RGB video frames are fed as inputs to a human motion segmentation algorithm to generate binary segmentation of human actions; at this point these human silhouettes do not carry any class- and instance-aware labels, and they only have binary labels for foreground (and the pixels don't belonging to human silhouettes are labelled as background class).* ***(b)*** *Our region proposal generation algorithm accepts the binary segmented video frames as inputs and computes region proposal bounding boxes using all possible combinations of 2D connected components* $(2^N - 1)$ *present in the binary map.* ***(c)*** *Once the region proposals are computed, warped regions are extracted from both RGB and optical flow frames and fed as inputs to the respective appearance- and motion-based detection networks.* ***(d)*** *The detection networks compute fc7 appearance and motion features for each warped region, features are then fused and subsequently used by a set of one-vs-all SVMs to generate action classification scores for each region.* ***(e)*** *Finally, frame-level detection windows are temporally linked as per their class-specific scores and spatial overlaps to build class-specific action tubes. Further, each pixel within the detection windows is assigned to an class- and instance-aware label by by utilising both the bounding-box detections associated with each class-specific action tubes and the binary segmentation maps (or human silhouettes) generated in* ***(a)****.*

Human motion segmentation. The human motion segmentation algorithm takes as input a sequence of RGB video frames (which contain human action) and outputs binary-labelled space-time video segments where pixels belong to an human action are labelled as foreground and remaining are as background.

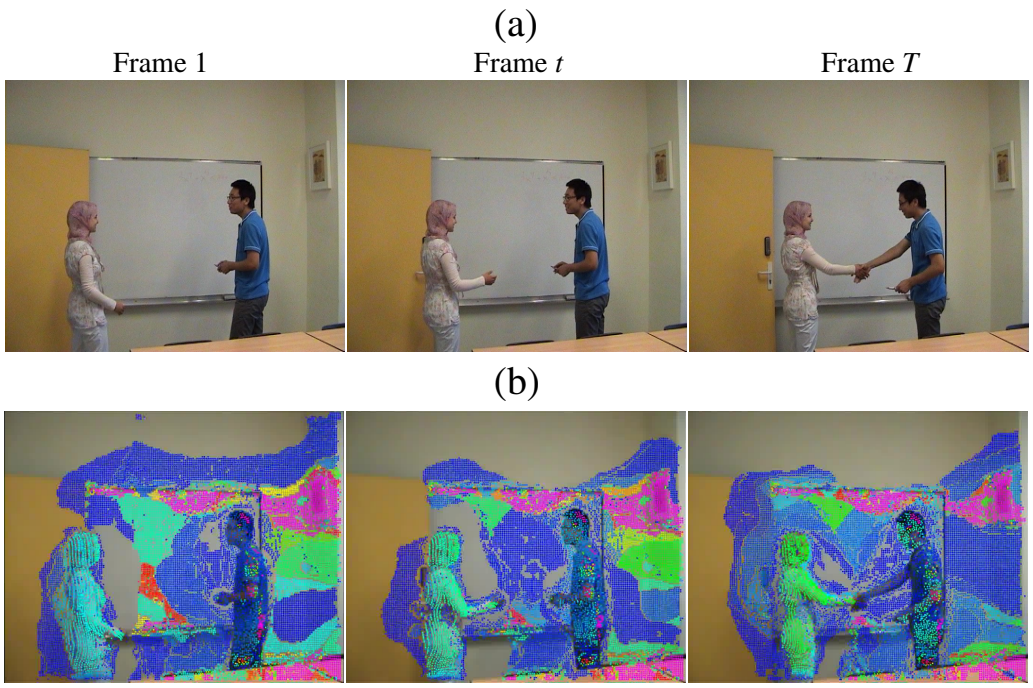


Figure 6.3: ***(a)*** *Three sample input video frames showing a "handshaking" action from a test video clip of LIRIS HARL dataset [128].* ***(b)*** *The corresponding motion saliency response generated using long term trajectories [39] are shown for these three frames. Notice, the motion saliency is realtively higher for the person at the left, who first enters into the room and then approaches towards the person in the right for "handshaking". Also note that, motion saliency is computed on the entire video clip, for the sake of visualization, we pick three sample frames.*

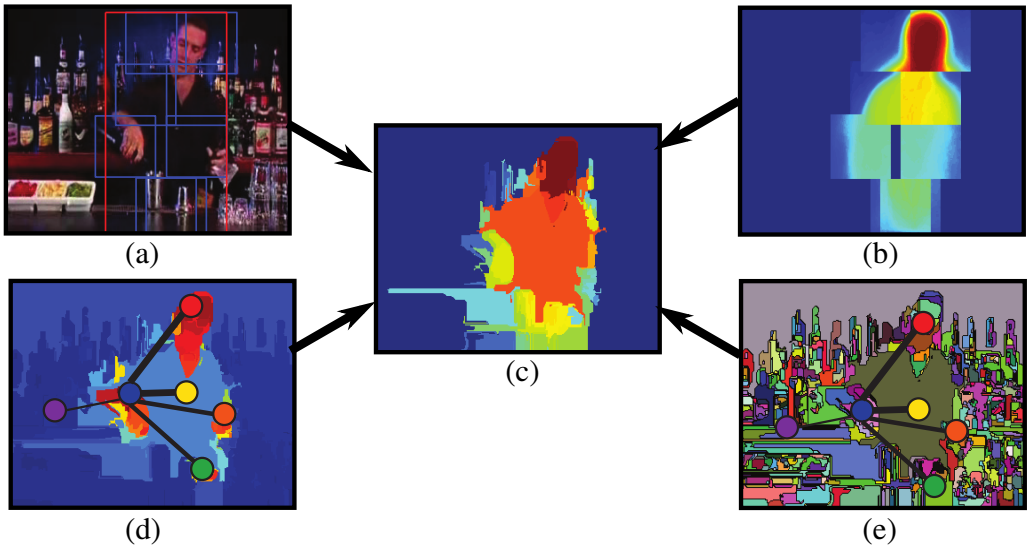


Figure 6.4: ***(a)*** *DPM based person detection.* ***(b)*** *Corresponding DPM part mask.* ***(c)*** *Supervoxel response for the DPM mask.* ***(d)*** *and* ***(e)*** *pairwise connections of motion saliency map and segmentation respectively. This figure is taken from [31] with author's permission.*

Firstly, in order to localise and rank "actionness" [51], a human motion saliency feature is computed by exploiting the foreground motion and human appearance information. Foreground motion is estimated by forming a camera model using long term trajectories [39] (Figure 6.3) and human appearance based saliency map is generated using a DPM person detector [91] (Figure 6.4 **(a-c)**) trained on PASCAL VOC 2007 [154]. Secondly, to segment human actions, a hierarchical graph-based video segmentation algorithm [155] is used to extract supervoxels at different level of pixel granularity (i.e. different levels of segmentation hierarchy) (Figure 6.5). The foreground motion and human appearance based saliency features are then encoded in the hierarchy of supervoxels using a hierarchical Markov Random Field (MRF) model. This encoding gives the unary potential components. To avoid a brittle graph due to a large number of supervoxels [132], the MRF graph is built with a smaller subset of supervoxels which are highly likely to contain human actions. Thus, a candidate edge is built between two neighbouring supervoxels based on their optical flow directions and overlaps with a person detection. In the MRF graph structure, supervoxels are nodes and an edge between two supervoxels are built if: (a) they are temporal neighbours i.e. neighbours in the direction of optical flow, or (b) spatial neighbours, i.e. both the supervoxels have high overlaps with a DPM person detection where the person detection has a confidence greater than a threshold. The temporal supervoxel neighbours and the appearance-aware spatial neighbours (Figure 6.4 **(d)** & **(e)**) give the pairwise potential components. To avoid leaks and encourage better semantic information, supervoxels (constrained by appearance and motion cues) from higher levels in the hierarchy (Figure 6.5) are supported by the higher-order potential. Finally, the energy of the MRF is minimised using the α-expansion algorithm [156, 157] and GMM estimation is used to automatically learn the model parameters. The final outputs of the human motion segmentation are the human foreground background binary maps as depicted in Figure 6.6.

Proposal based on Selective Search. We use two competing approaches to generate region proposals for action detection. The first is based upon Selective Search [47], and the second approach is presented in Section 6.3.1. Whilst using the Selective Search based method for both training and testing, we only use the motion segmentation based method for testing since it does not provide good negative proposals to use during training. Having a sufficient number of negative examples is crucial to train an effective classifier. At test time, the human motion segmentation (Section 6.3.1) allows us to extract pixel-level action

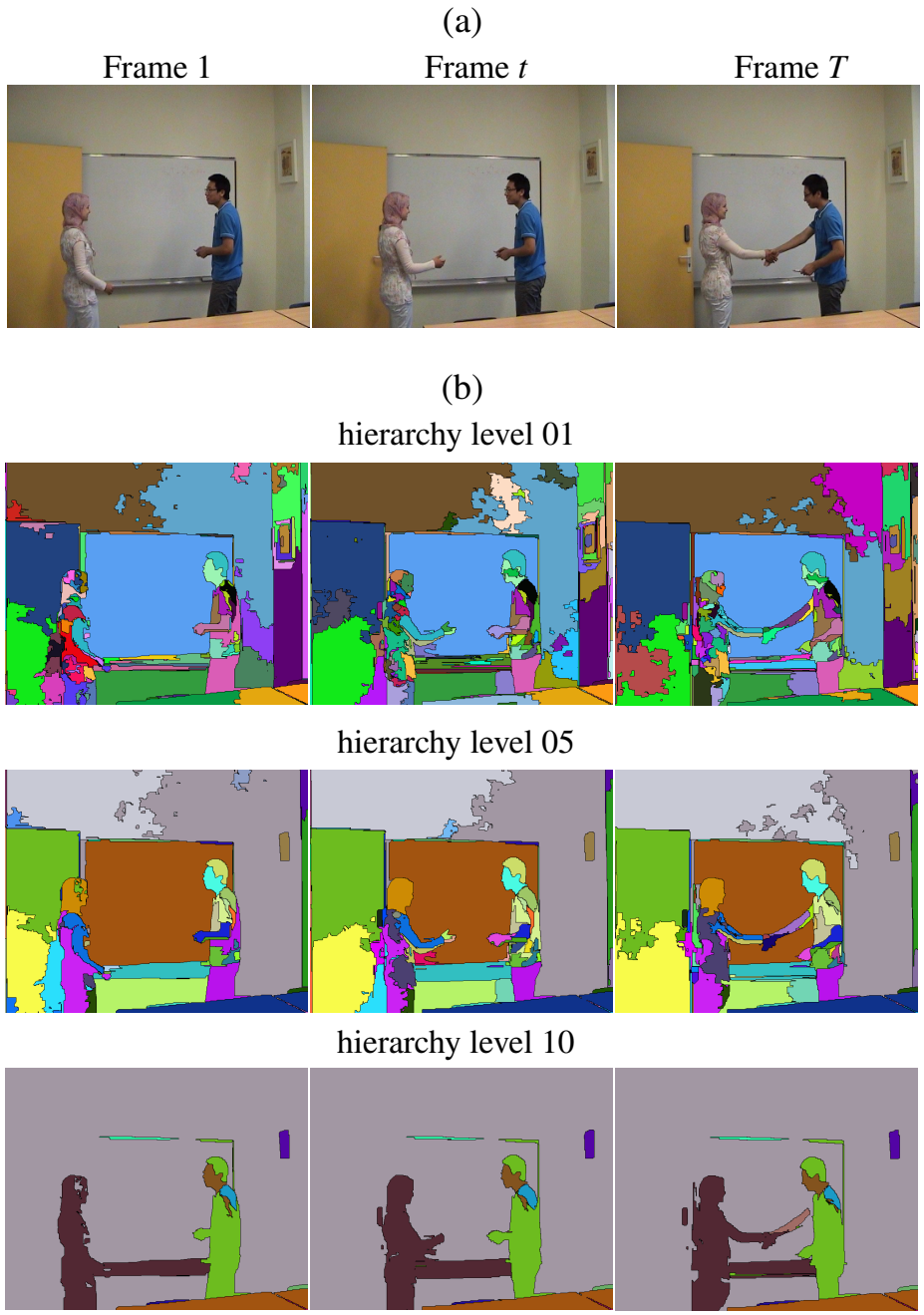


Figure 6.5: ***(a)*** *Three sample input video frames showing a "handshaking" action from a test video clip of LIRIS HARL dataset [128].* ***(b)*** *The hierarchical graph based video segmentation results (at three different levels of hierarchy) are shown for these three frames. The three rows show segmentation results for hierarchy level* 1, 5 *and* 10 *respectively where* 1 *is the lowest level with supervoxels having smaller spatial extents and* 10 *is the highest level with supervoxels having relatively larger spatial extents. Notice, the supervoxels belong to higher levels of segmentation hierarchy tend to preserve the semantic information and are less prone to leaks. Also note that, video segmentation is computed on the entire video clip, for the sake of visualization, we pick three sample frames.*

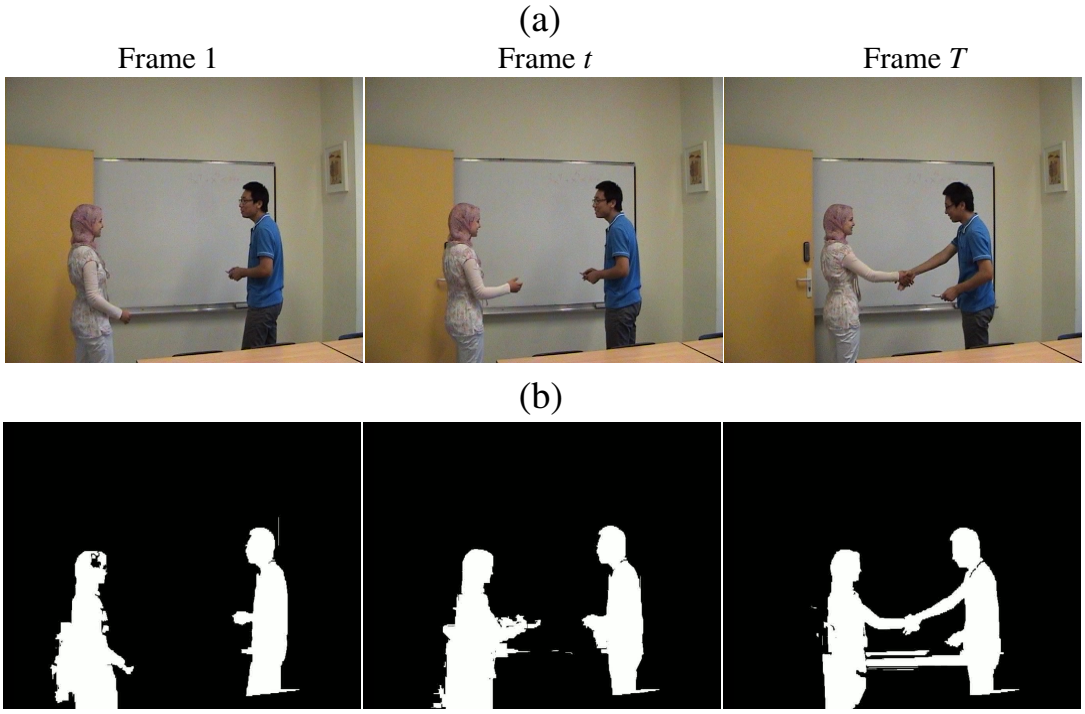


Figure 6.6: ***(a)*** *Three sample input video frames showing a "handshaking" action from a test video clip of LIRIS HARL dataset [128].* ***(b)*** *The human action foreground-background segmentation results are shown for these three frames.*

instance segmentation which is superior to what we may obtain by using Selective Search. We validate our action detection pipeline using both algorithms - the results are discussed in Section 6.4.

Measuring "actionness" of Selective Search proposals. The selective-search region-merging similarity score is based on a combination of colour (histogram intersection), and size properties, encouraging smaller regions to merge early, and avoid holes in the hierarchical grouping. Selective Search (SS) generates on average 2,000 region proposals per frame, most of which do not contain human activities. In order to rank the proposals with an "actionness" score and prune irrelevant regions, we compute dense optical flow between each pair of consecutive frames using the state-of-the-art algorithm in [158]. Unlike Gkioxari and Malik [14], we use a relatively smaller motion threshold value to prune SS boxes, (Section A.4) to avoid neglecting human activities which exhibit minor body movements exhibited in the LIRIS HARL [128] such as "typing on keyboard", "telephone conversation" and "discussion" activities. In addition to pruning region proposals, the 3-channel optical flow values (i.e., flow-x, flow-y and the flow magnitude) are used to construct 'motion images' from which CNN

motion features are extracted [14].

6.3.2 Appearance- and motion-based detection networks

In the second stage of the pipeline, we use the "actionness" ranked region proposals (Section 6.3.1) to select image patches from both the RGB (original video frames) and flow images. The image patches are then fed to a pair of fine-tuned Convolutional Neural Networks (Figure 6.2 **(d)**) (which encode appearance and local image motion, respectively) from which appearance and motion feature vectors were extracted. As a result the first network learns static appearance information (both lower-level features such as boundary lines, corners, edges and high level features such as object shapes), while the other encodes action dynamics at frame level. The output of the Convolutional Neural Network may be seen as a highly nonlinear transformation $\Phi(.)$ from local image patches to a high-dimensional vector space in which discrimination may be performed accurately even by a linear classifier. We follow the AlexNet [92] and [159]'s network architectures.

Pretraining. We adopt a CNN training strategy similar to [28]. Indeed, for domain-specific tasks on relatively small scale datasets, such as LIRIS HARL [128], it is important to initialise the CNN weights using a model *pre-trained on a larger-scale dataset*, in order to avoid over-fitting [14]. Therefore, to encode object "context" we initialise the appearance-based CNN's weights using a model pre-trained on the PASCAL VOC 2012's object detection dataset. To encode typical motion patterns over a temporal window, the optical motion-based CNN is initialised using a model pre-trained on the UCF101 dataset (split 1) [17]. Both appearance- and motion-based pre-trained models are publicly available online at `https://github.com/gkioxari/ActionTubes`.

Fine tuning. We use deep learning software tool Caffe [160] to fine-tune pre-trained domain-specific appearance- and motion-based CNNs on LIRIS HARL training set. For training CNNs, the Selective Search region proposals (Section 6.3.1) with an IoU overlap score greater than 0.5 with respect to the ground truth bounding box were considered as positive examples, the rest as negative examples. The image patches specified by the pruned region proposals were randomly cropped and horizontally flipped by the Caffe's *WindowDataLayer* [160] with a crop dimension of 227×227 and a flip probability of 0.5 (Figure 6.2 **(c)**). Random cropping and flipping were done for both RGB and flow images. The

pre-processed image patches along with the associated ground truth action class labels are then passed as inputs to the appearance and motion CNNs to fine-tune (i.e. updating only the weights of the fully connected layers, in this case, fc6 and fc7 layers, and keeping the weights of the other layers untouched during training) for action classification (Figure 6.2 **(d)**). A mini batch of 128 image patches (32 positive and 96 negative examples) are processed by the CNNs at each training forward-pass. Note that the number of batches varies frame-to-fame as per the number of ranked proposals per frame. It makes sense to include fewer positive examples (action regions) as these are relatively rare when compared to background patches (negative examples).

Feature extraction from CNN layers. We extract the appearance- and motion-based features from the *fc7* layer of the the two networks. Thus, we get two feature vectors (each of dimension 4096): appearance feature '$\mathbf{x}_a = \Phi_a(\mathbf{r})$' and motion feature '$\mathbf{x}_f = \Phi_f(\mathbf{r})$'. We perform L2 normalisation on the obtained feature vectors, to then, scale and merge appearance and motion features (Figure 6.2 **(d)**) in an approach similar to that proposed by [14]. This yields a single feature vector $\mathbf{x}$ for each image patch $\mathbf{r}$. Such frame-level region feature vectors are used to train an SVM classifier (Section 6.3.3).

6.3.3 Training region proposal classifiers

Once discriminative CNN fc7 feature vectors $\mathbf{x} \in \mathbb{R}^n$ are extracted for region proposals (Section 6.3.1), they can be used to train a set of binary classifiers (Figure 6.2 **(d)**) to attach a vector of scores $\mathbf{s}_c$ to each region proposal '$\mathbf{r}$', where each element in the score vector $\mathbf{s}_c$ is a confidence measure of each action class $c \in \{1, 2, \ldots, C\}$ to be present within that region. Due to the notable success of linear SVM classifiers when combined with CNN features [28], we trained a set of 1-vs-rest linear SVMs to classify region proposals.

Class specific positive and negative examples. In the original RCNN-based one-vs-rest SVM training approach [28], only the ground truth bounding boxes are considered as positive training examples. In contrast, due to extremely high inter- and intra-class variations in LIRIS HARL dataset [128], we use those bounding boxes as positive training examples which have an IoU overlap with the ground truth greater than 75%. In addition, we also consider the ground truth bounding boxes as positives. We believe, our this training data sampling scheme is more intuitive for complex datasets to train SVMs with more positive

examples rather than only ground truths. We have achieved almost 5% gain over SVMs classification accuracy with this training strategy. In a similar way, we consider as negative examples only those features vectors whose associated region proposal have an overlap smaller than 30% with respect to the ground truth bounding boxes (possibly several) present in the frame.

Training with hard negative mining. We train the set of class specific linear SVMs using hard negative mining [91] to speed up the training process. Namely, in each iteration of the SVM training step we consider only those negative features which fall within the margin of the decision boundary. We use the publicly available toolbox *Liblinear*[1] for SVM training and use $L2$ regularizer and $L1$ hinge-loss with the following parameter values to train the SVMs: positive loss weight $W_{LP} = 2$; SVM regularisation constant $C = 10^{-3}$; bias multiplier $B = 10$.

6.3.4 Testing region proposal classifiers

With our actionness-ranked region proposals $\mathbf{r}_\text{i}$ (§ 6.3.1) we can extract a cropped image patch and pass it to the CNNs for feature extraction in a similar fashion as described in Section 6.3.2. A prediction takes the form:

$$\mathbf{s}_c(\mathbf{b}) = \mathbf{w}_c^T \Phi(\mathbf{r}) + b_c^{svm}, \tag{6.1}$$

where, $\Phi(\mathbf{r}) = \{\Phi_a(\mathbf{r}); \Phi_f(\mathbf{r})\}$ is combination of appearance and motion features of $\mathbf{r}$, $\mathbf{w}_c^T$ and b_c^{svm} are the hyperplane parameter and the bias term of the learned SVM model of class c. The confidence measure $\mathbf{s}_c(\mathbf{b})$ that the action 'c' has happened within the bounding-box region '$\mathbf{b}$' is based on the appearance and motion features. Here $\mathbf{b}$ denotes the associated bounding box for a region proposal $\mathbf{r}$.

After SVM prediction, each region proposal '$\mathbf{r}$' has been assigned a set of class-specific scores $\mathbf{s}_c$, where c denotes the action category label, $c \in \{1, \ldots, C\}$. Once a region proposal has been assigned classification scores $\mathbf{s}_c$, we call it as a detection bounding-box and denote it as $\mathbf{b}$. Due to the typically large number of region proposals generated by the Selective Search algorithms (§ 6.3.1), we further apply non-maximum suppression to prune the regions.

[1] `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`.

6.3.5 Action tube generation and classification

Once we extract the frame-level detection boxes $\mathbf{b}_t$ (§ Section 6.3.4) for an entire video, we would like to identify sequences of detections most likely to form action tubes. Thus, to extract final detection tubes, linking of these detection boxes in time is essential to generate tubes. We use our two-pass dynamic programming approach presented in Section 5.3.4 to formulate the action tube generation problem as a labelling problem where: i) we link detections $\mathbf{b}_t$ into temporally connected action paths for each action, and ii) we perform a piece-wise constant temporal labelling on the action paths. A detailed formulation of the tube generation problem is presented in Section A.5.

6.4 Experimental results

We evaluate two region proposal methods with our pipeline, one based on human motion segmentation (HMS) (Section 6.3.1) and another one based on selective search (SS) (Section 6.3.1). We will use "HMS" and "SS" abbreviations in tables and plot to show the performance of our pipeline based on each region proposal technique. Our results are also compared to the current state-of-the-art: VPULABUAM-13 [143] and IACAS-51 [144].

6.4.1 Instance classification performance - no localisation (NL)

This evaluation strategy ignores the localisation information (i.e. the bounding boxes) and only focuses on whether an action is present in a video or not. If a video contains multiple actions then system should return the labels of all the actions present correctly. Even though our action detection framework is not specifically designed for this task, we still outperform the competition, as shown in Table 6.1.

6.4.2 Detection and localisation performance

This evaluation strategy takes localisation (space and time) information into account [151]. We use a 10% threshold quality level for the four thresholds (Section 4.2.5), which is the same as that used in the LIRIS-HARL competition. In Table 6.1, we denote these results as "method-name-NL" (NL for no localisation) and "method-name-10%". In both cases (without localisation and

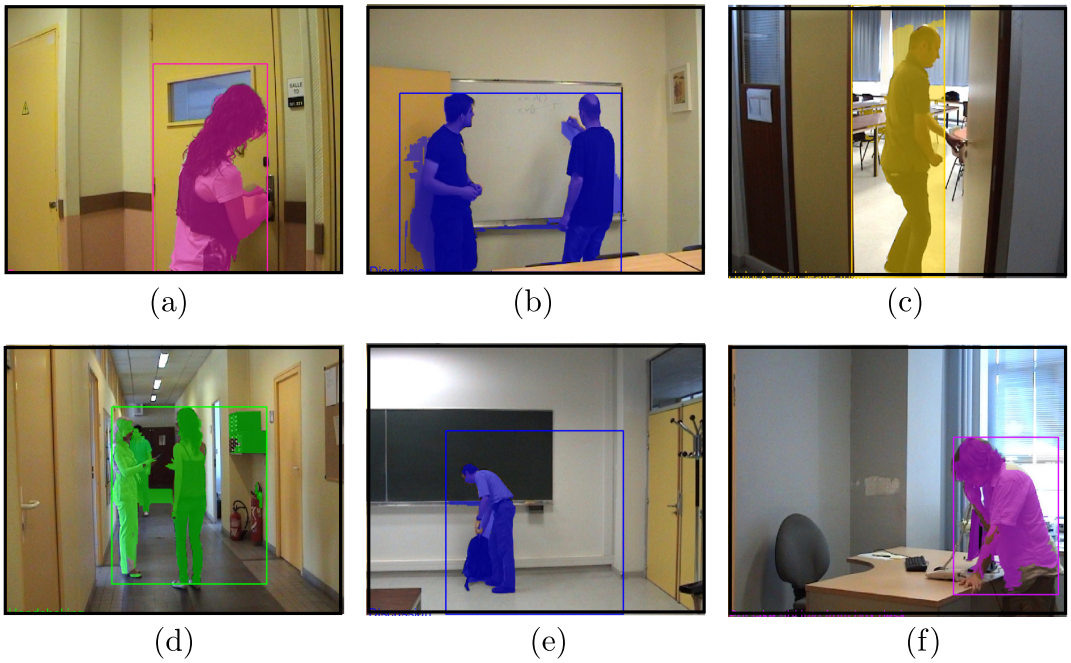


Figure 6.7: Correct (a-c) and incorrect (d-f) instance segmentation results on the LIRIS-HARL dataset [128], the correct category is shown in brackets. ***(a)*** *'Try enter room unsuccessfully'.* ***(b)*** *'Discussion'.* ***(c)*** *'Unlock enter/leave room'.* ***(d)*** *'Hand-shaking' (Give take object from person).* ***(e)*** *'Discussion' (Leave bag unattended).* ***(f)*** *'Put take object into/from desk' (Telephone conversation).*

with 10% overlap), our method outperforms existing approaches, achieving an improvement from 46% [143] to 56%, in terms of F1 score without localisation measures, and a improvement from 5% [143] to 56% (11.2 times better) gain in the F1-score when 10% localisation information *is* taken into account. In Table 6.2 we list the results we obtained using the overall integrated performance scores (Equation 4.11) - our method yields significantly better quantitative and qualitative results with an improvement from 3% [143] to 43% (14.3% times better) in terms of F1 score, a relative gain across the spectrum of measures. Samples of qualitative instance segmentation results are shown in Figure 6.7.

The pure classification accuracy of the HMS- and SS-based approaches are reflected in the Confusion Matrices shown in Figure 6.9. Confusion matrices show the the complexity of the dataset. Some of the actions are wrongly classified, e.g., *"telephone-conversation"* is classified as *"put/take object to/from box/desk"*, same can be observed for action *"unlock enter/leave room"* in SS approach.

Method	Recall	Precision	F1-Score
VPULABUAM-13-NL	0.36	**0.66**	0.46
IACAS-51-NL	0.3	0.46	0.36
SS-NL (ours)	**0.5**	0.53	**0.52**
HMS-NL (ours)	**0.5**	0.63	**0.56**
VPULABUAM-13-10%	0.04	0.08	0.05
IACAS-51-NL-10%	0.03	0.04	0.03
SS-10% (ours)	**0.5**	0.53	0.52
HMS-10% (ours)	**0.5**	**0.63**	**0.56**

Table 6.1: Quantitative measures precision and recall on LIRIS HARL dataset.

Method	I_{sr}	I_{sp}	I_{tr}	I_{tp}	IQ
VPULABUAM-13-IQ	0.02	0.03	0.03	0.03	0.03
IACAS-51-IQ	0.01	0.01	0.03	00.0	0.02
SS-IQ (ours)	0.52	0.22	0.41	0.39	0.38
HMS-IQ (ours)	0.49	0.35	0.46	0.43	**0.44**

Table 6.2: Qualitative thresholds and integrated score on LIRIS HARL dataset.

6.4.3 Performance vs detection quality curves

The plots in Figure 6.8 attest the robustness of our method, as they depict the curves corresponding to precision, recall and F1-score over varying quality thresholds.

When the threshold t_{tr} for temporal recall is considered (see Figure 6.8 plot-(a)) we achieved a highest recall of 50% for both HMS- and SS-based approaches and a highest precision of 65% for HMS-based approach at threshold value of t_{tr}=0. As the threshold increases towards $t_{tr} = 1$, SS-based method shows a robust performance, with highest recall=50% and precision=52%, HMS-based method shows promising results with an acceptable drop in precision and recall. Note that when t_{tr}=1, we assume that all frames of an activity instance need to be detected in order for the instance itself to be considered as detected.
As for the competing methods, IACAS-51 [144] yields the next competing recall of 2.4% and a precision of 3.7% with a threshold value of t_{tr}=1.

When acting on the value of the temporal frame-wise precision threshold t_{tp} (see Figure 6.8 plot-(b)) we can observe that at t_{tp}=1, when we assume that not a single spurious frame outside the ground truth temporal window is allowed, our HMS-based region proposal approach gives highest recall of 8% and precision 10.7%, where, as SS-based approach has significantly lower recall=2%

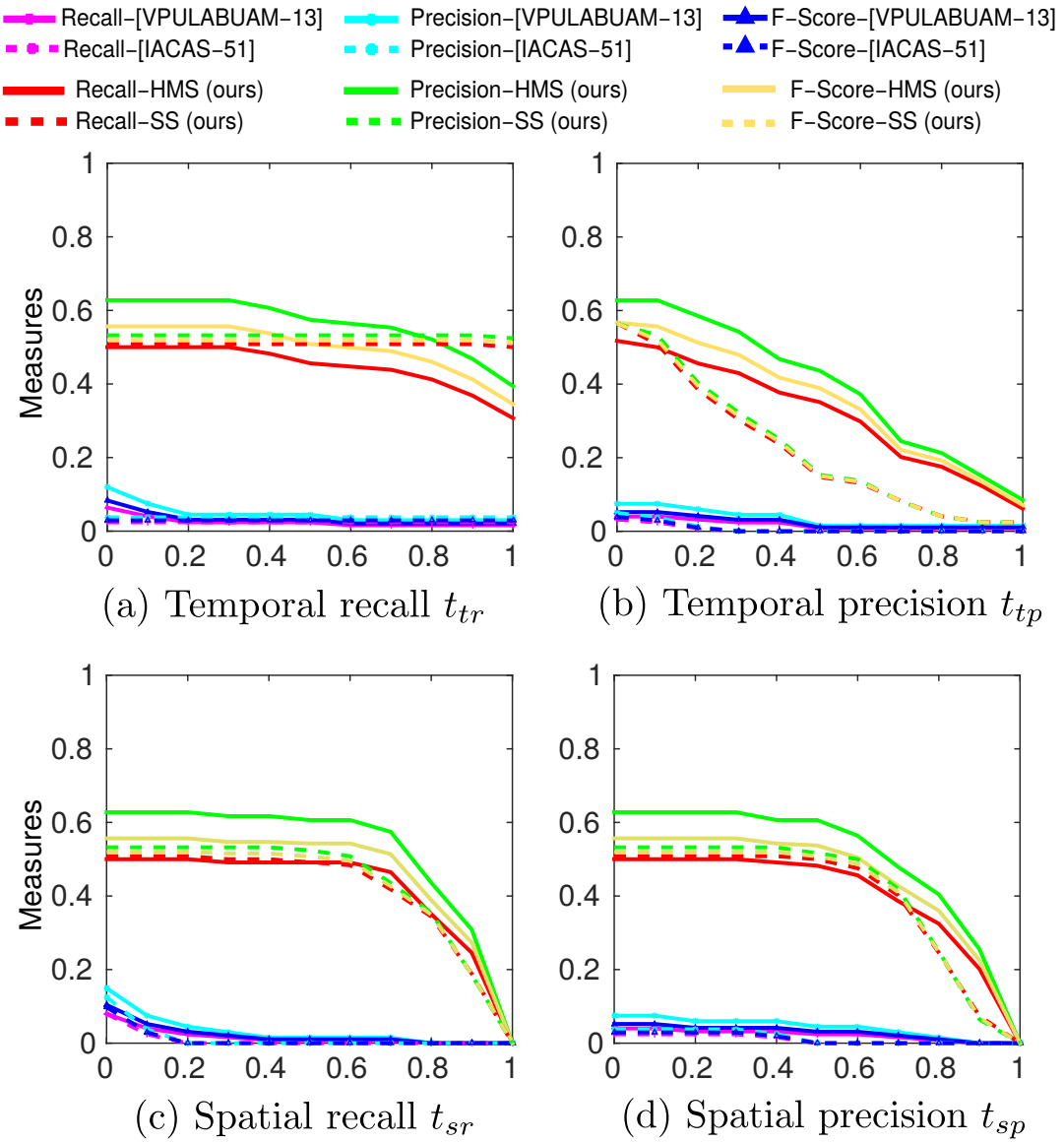


Figure 6.8: Performance vs detection quality curves.

and precision=2.4%, which is still significantly higher than the performance of the existing methods. Indeed, at t_{tp}=1, VPULABUAM-13 has recall=0.8% and precision=1% where IACAS-51 yields both zero precision and zero recall. This results tell us that HMS-based approach performs superior in detecting temporal extent of an action and thus is suitable for action localisation in temporally untrimmed videos. The remaining two plots-(c) and -(d) of Figure 6.8 illustrate the overall performance when spatial overlap is taken into account. Both plots show metrics approaching zero when the corresponding spatial thresholds (pixel-wise recall t_{sr} and pixel-wise precision t_{sp}) approach 1. Note that it is highly unlikely for a ground truth activity to be consistently (spatially) included in the

corresponding detected activity over all the consecutive frames (spatial recall), as indicated in the plot-(c). It is also rare for a detected activity to be (spatially) included in the corresponding ground truth activity over all the frames (spatial precision) as indicated in plot-(d).
For the pixel-wise recall (plot-(c)), our HMS based method shows consistent recall between 45% to 50% and precision between 59% to 65.5% up to a threshold value of t_{sr}=0.7, where as, SS-based region proposal approach gives comparable recall between 48.3% to 50.8%, but relative lower precision between 43.5% to 53.2% up to t_{sr}=0.7. For the pixel-wise precision (plot-(d)), HMS and SS-based approaches give similar recall between 39% to 50%, where as HMS-method again outperforms in precision with 48% to 63% up to a threshold value of t_{sp}=0.7, where as SS has precision 41% to 53% up to a threshold value t_{sp}=0.7. Finally, we draw conclusion that our HMS-based region proposal approach shows superior qualitative and quantitative detection performance on the challenging LIRIS HARL dataset.

6.4.4 Qualitative action instance segmentation and localisation results

LIRIS HARL dataset. Figure 6.10 shows additional qualitative action instance segmentation and localisation results on LIRIS HARL dataset [128]. In particular, Figure 6.10 **(1)** and **(4)** show that the proposed approach can successfully detect action instances belonging to a same class or different classes at finer pixel-level. In **(1)**, two action instances of a single action class (i.e. *"typing on keyboard"*) are present, whereas in **(4)** two action instances belonging to two different action classes (i.e. *"handshaking"* and *"leave baggage unattended"*) are present.

UCF-101-24 dataset. To demonstrate that the proposed instance segmentation method generalises well on other datasets, we present here some sample instance segmentation results on UCF-101-24. We compute the binary segmentation masks for some selected UCF-101-24 test video clips, and apply the bounding-boxes predicted by the model proposed in Chapter 5 on the top of the binary masks to generate the final instance segmentation results which are shown in Figure 6.11 and 6.12. Note that, the proposed approach can successfully localise multiple instances of the *"biking"* (Figure 6.11 **(b)**), *"fencing"* (Figure 6.12 **(a)**), and *"ice dancing"* (Figure 6.12 **(c)**) actions at finer pixel level

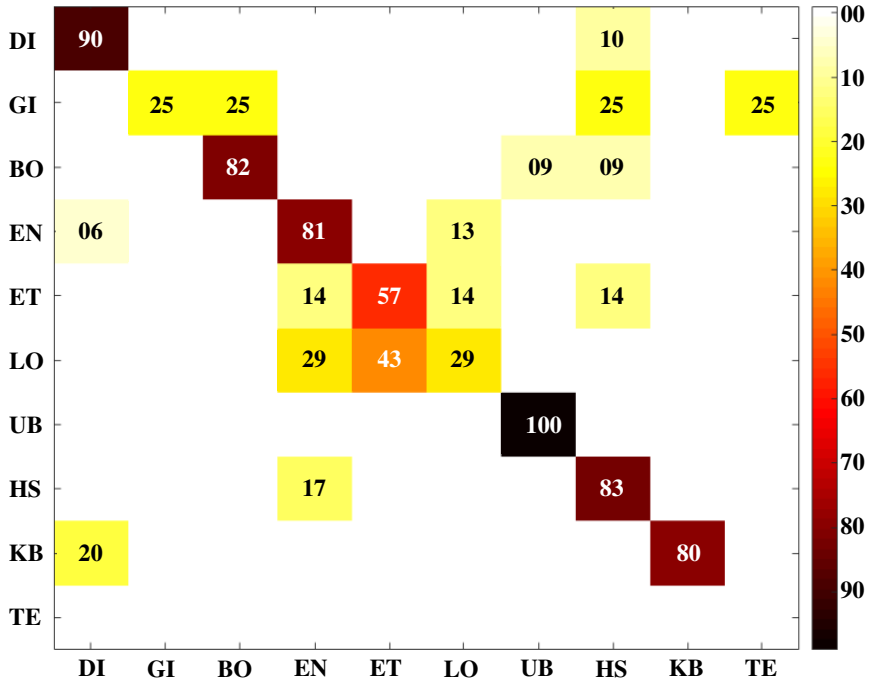
in space and time.

6.5 Discussion

Unlike state-of-the-art supervised instance segmentation approaches (for objects) [138, 161] which require expensive ground-truth segmentation (i.e. per pixel class- and instance-aware labelling) to train their networks, the proposed framework does not require such expensive ground-truth annotations. Thanks to the human action segmentation [31] algorithm which computes human action binary masks using unsupervised learning, thus, does not require expensive ground-truth labels. However, the major drawback of [31] is that it is computationally expensive. For example, it takes several days to compute the binary masks for all frames in LIRIS HARL dataset. Another limitation is that the HMS (human motion segmentation) based region proposals fail to generate accurate bounding box proposals in cases where the action segmentations of two or multiple actors get merged into one 2D connected component, e.g., see Figure 6.10 **(8)** in which out of two instances of *"typing on keyboard"* action class, only one instance has been successfully detected. We empirically found that in such instances Selective Search based region proposals work more effectively. Lastly, as there are no ground truth instance segmentation annotations available for LIRIS HARL and UCF-101-24 datasets, we could not perform an quantitative evaluation of the instance segmentation results. Also note, the J-HMDB-21 dataset has a single action instance per video, and thus, not suitable for evaluating instance segmentation methods.

[**DI**]-discussion [**GI**]-give object to person [**BO**]-put/take obj into/from box/desk [**HS**]-handshaking
[**EN**]-enter/leave room no unlocking [**ET**]-try enter room unsuccessfully [**LO**]-unlock enter leave room
[**UB**]-leave baggage unattended [**KB**]-typing on keyboard [**TE**]-telephone conversation

(a) HMS region proposals



(a) SS region proposals

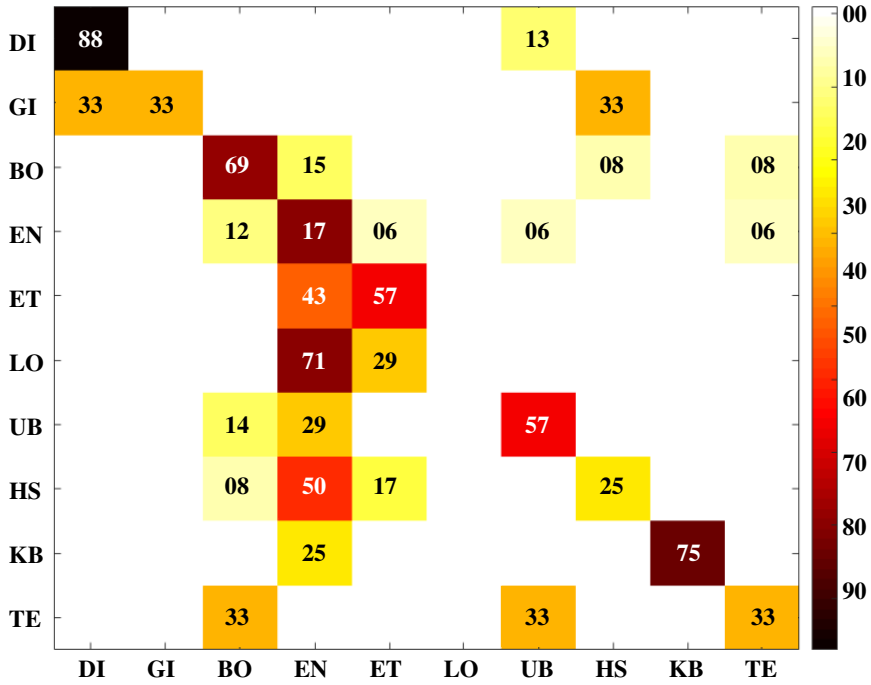


Figure 6.9: Confusion matrix obtained by human motion segmentation(HMS) and selective search(SS) region proposal approach. They show the classification accuracy of HMS- and SS-based methods on LIRIS HARL human activity dataset. HMS region proposal based method provides better classification accuracy on the the complex LIRIS dataset [128].

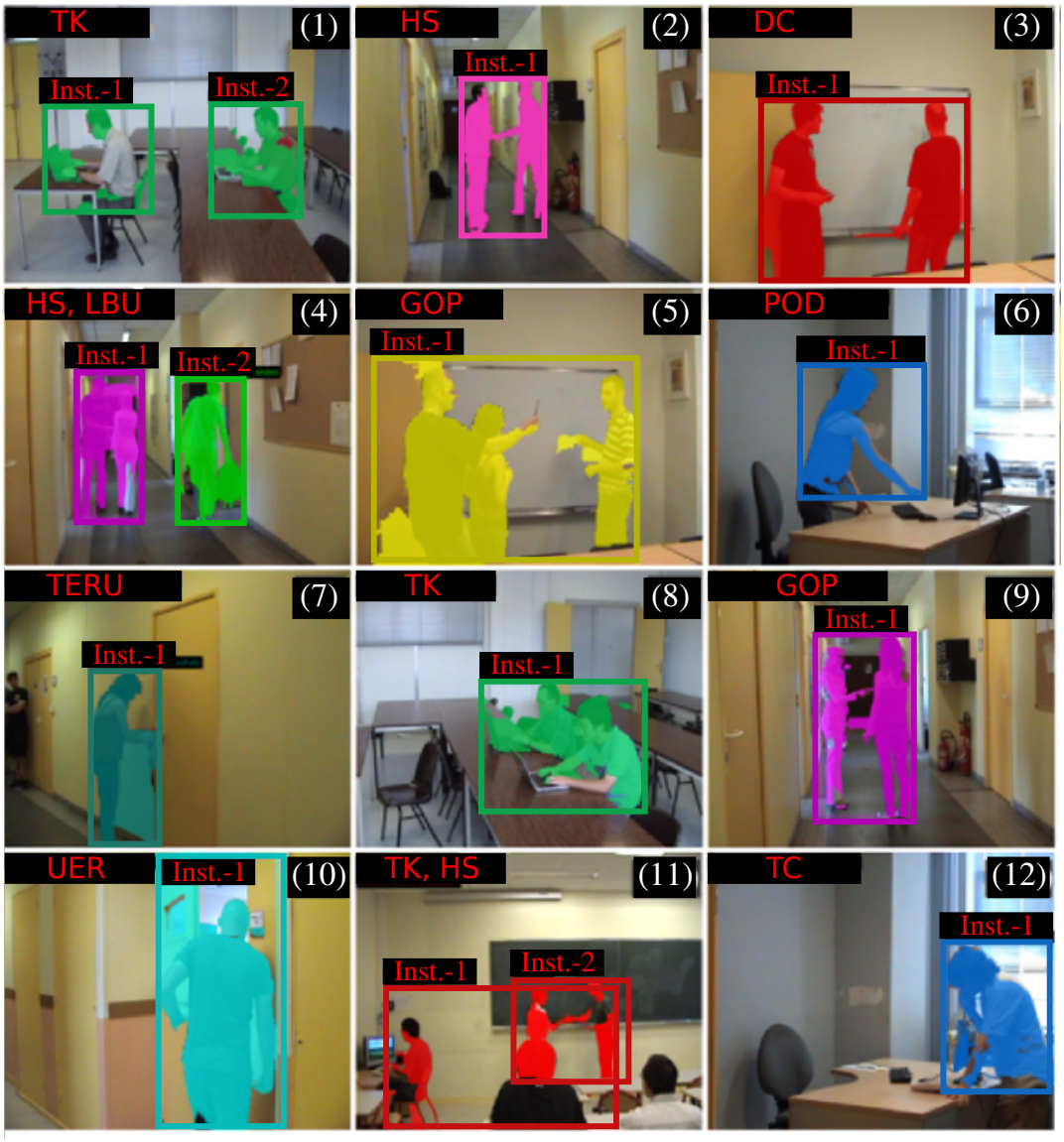


Figure 6.10: Qualitative action instance segmentation and localisation results on LIRIS HARL dataset. Ground-truth action labels: ***TK*** *– typing on keyboard,* ***HS*** *– handshaking,* ***DC*** *– discussion,* ***LBU*** *– leave baggage unattended,* ***GOP*** *– give object to person,* ***POD*** *– put object into desk,* ***TERU*** *– try enter room unsuccessfully,* ***UER*** *– unlock enter room,* ***TC*** *– telephone conversation. Correct results:* ***(1)****,* ***(2)****,* ***(3)****,* ***(4)****,* ***(5)****,* ***(6)****,* ***(7)****,* ***(8)****,* ***(10)****; incorrect results:* ***(8)****,* ***(9)****,* ***(11)****,* ***(12)****. In* ***(8)****, out of two instances of* ***TK*** *action class, only one instance has been successfully detected. In* ***(9)****, the ground truth action class* ***GOP*** *has been misclassified as* ***HS*** *class. In* ***(11)****, the ground truth action classes* ***TK*** *and* ***HS*** *have been misclassified as* ***DC*** *class. In* ***(12)****, the ground truth action class* ***TC*** *has been misclassified as* ***POD*** *class.*

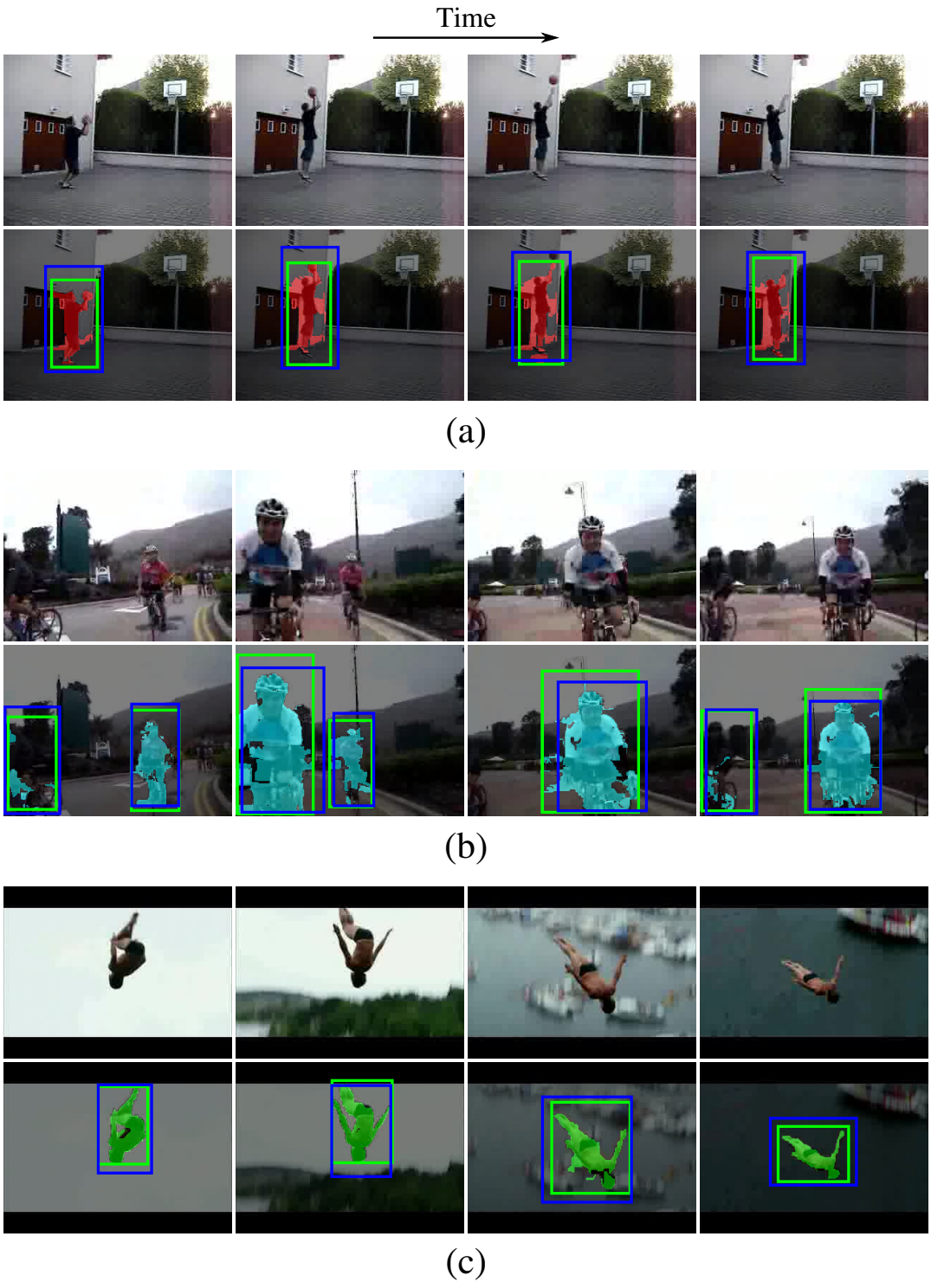


Figure 6.11: Qualitative action instance segmentation and localisation results on UCF-101-24 test videos. The green boxes represent ground truth annotations, whereas the blue boxes denote the frame-level detections. Each row represents an UCF-101-24 test video clip where the 1^{st} and 2^{nd} rows in each set (i.e. set ***(a)****,* ***(b)*** *and* ***(c)****) are the input video frames and their corresponding outputs respectively. From each clip 4 selected frames are shown. Predicted action labels:* ***(a)*** *"basketball";* ***(b)*** *"biking";* ***(c)*** *"cliffdiving".*

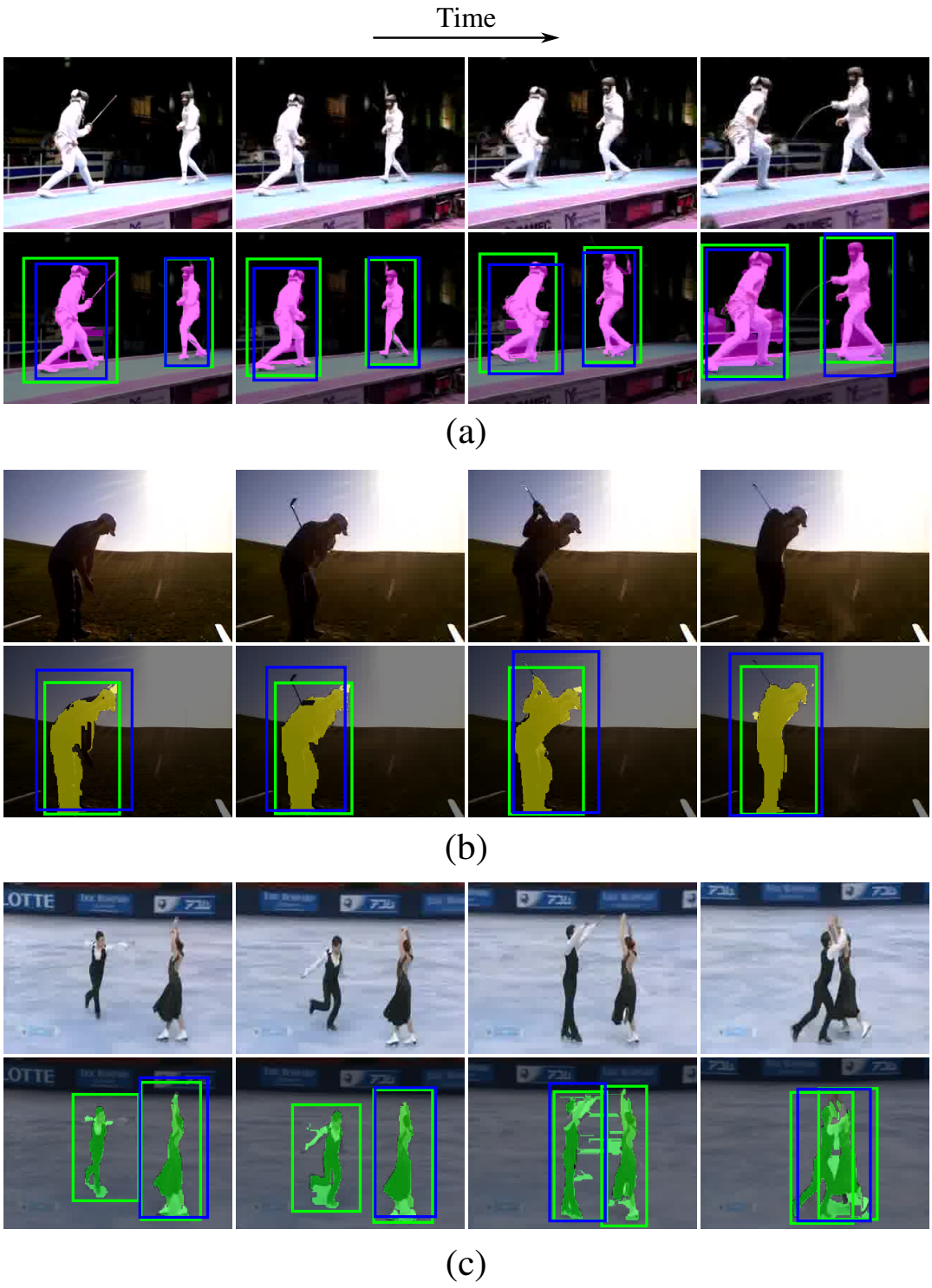


Figure 6.12: Qualitative action instance segmentation and localisation results on UCF-101-24 test videos. The green boxes represent ground truth annotations, whereas the blue boxes denote the frame-level detections. Each row represents an UCF-101-24 test video clip where the 1^{st} and 2^{nd} rows in each set (i.e. set ***(a)****,* ***(b)*** *and* ***(c)****) are the input video frames and their corresponding outputs respectively. From each clip 4 selected frames are shown. Predicted action labels:* ***(a)*** *"fencing";* ***(b)*** *"golfswing";* ***(c)*** *"icedancing".*

Chapter 7

AMTnet: Action-Micro-Tube Regression Using Deep Architecture

7.1 Introduction

In previous chapters (Chapter 5 and Chapter 6), we present action detection frameworks which are based on frame-level representation (Section 2.1) resulting a suboptimal solution to the action detection problem (Section 2.2). In this chapter, we propose a new deep network architecture which exploits the video-level representation and 3D region proposals (Section 2.3) to classify and regress whole video subsets. The work presented in this chapter radically departs from current practice of frame-level action representation, and take a first step towards a truly optimal solution of the action detection problem by considering video-level space-time action region hypotheses formed by a pair of bounding boxes spanning two successive video frames at an arbitrary temporal interval Δ (Section 2.3 , Figure 2.4 & Figure 7.1). We call these pairs of bounding boxes *3D region proposals*.

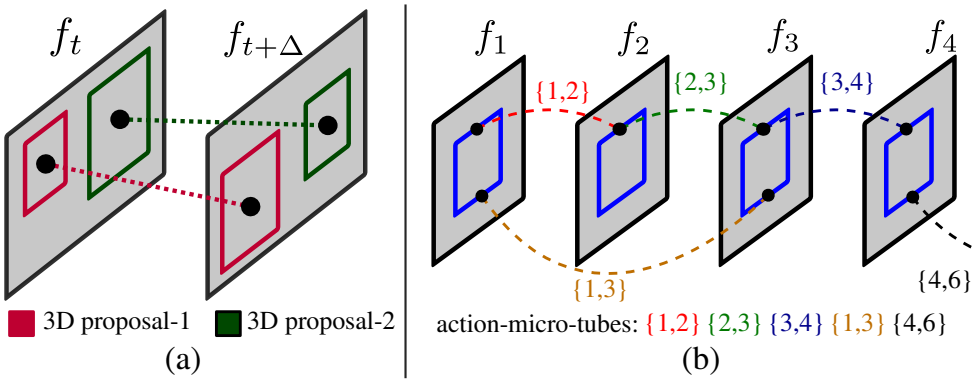


Figure 7.1: ***(a)*** *The 3D region proposals generated by our 3D-RPN network span pairs of successive video frames f_t and $f_{t+\Delta}$ at temporal distance Δ.* ***(b)*** *Ground-truth action-micro-tubes generated from different pairs of successive video frames.*

The advantages of this approach are that a) appearance features can be exploited to learn temporal dependencies (unlike what happens in current approaches), thus boosting detection performance; b) the linking of frame-level detections over time (i.e. solving the inter-frame data association problem (Section 1.3)) is no longer a post processing step and can be (partially) learned by the network. Obviously, at this stage we still need to construct action tubes from 3D region proposals.

We thus propose a radically new approach to action detection based on (1) a novel deep learning architecture for classifying space-time action regions and regressing 3D proposals (Section 2.3 & Figure 2.4) illustrated in Figure 7.2, in combination with (2) an original strategy for linking micro-tubes up into proper action tubes. At test time, this new framework does not completely rely on post-processing for assembling frame-level detections, but makes use of the temporal encoding learned by the network. We show that: i) such a network trained on pairs of successive RGB video frames can learn the spatial and temporal extents of action instances relatively better than those trained on individual video frames, and ii) our model outperforms the current state-of-the-art [14, 20, 33] in spatio-temporal action detection by just exploiting appearance (the RGB video frames), as opposed to the methods which heavily exploit expensive optical flow maps.

The aim of this chapter is not to renounce to optical flow cues, but to move from frame-level detections to whole tube regression. Indeed the method can be easily extended to incorporate motion at the micro-tube level rather than frame level, allowing fusion of appearance and motion at training time, unlike current methods [32, 33]. An extension of this work which incorporates optical flow signals is presented in Chapter 8.

Outline. This chapter is organized as follows. We present an overview of our approach in Section 7.2 and then give the details of the approach including network architecture, training and micro-tube linking in Sections 7.3, 7.4 and 7.5 respectively. Next, we report an extensive evaluation of the proposed model in Section 7.6. Section 7.7 presents the supporting experiments and discussion. Finally, in Section 7.8 we provide the implementation details.

Related publication. The work presented in this chapter has appeared in ICCV 2017 [38]. In Section 7.6 and 7.7, we refer the work presented in Chapter 5 with its associated BMVC 2016 publication [33].

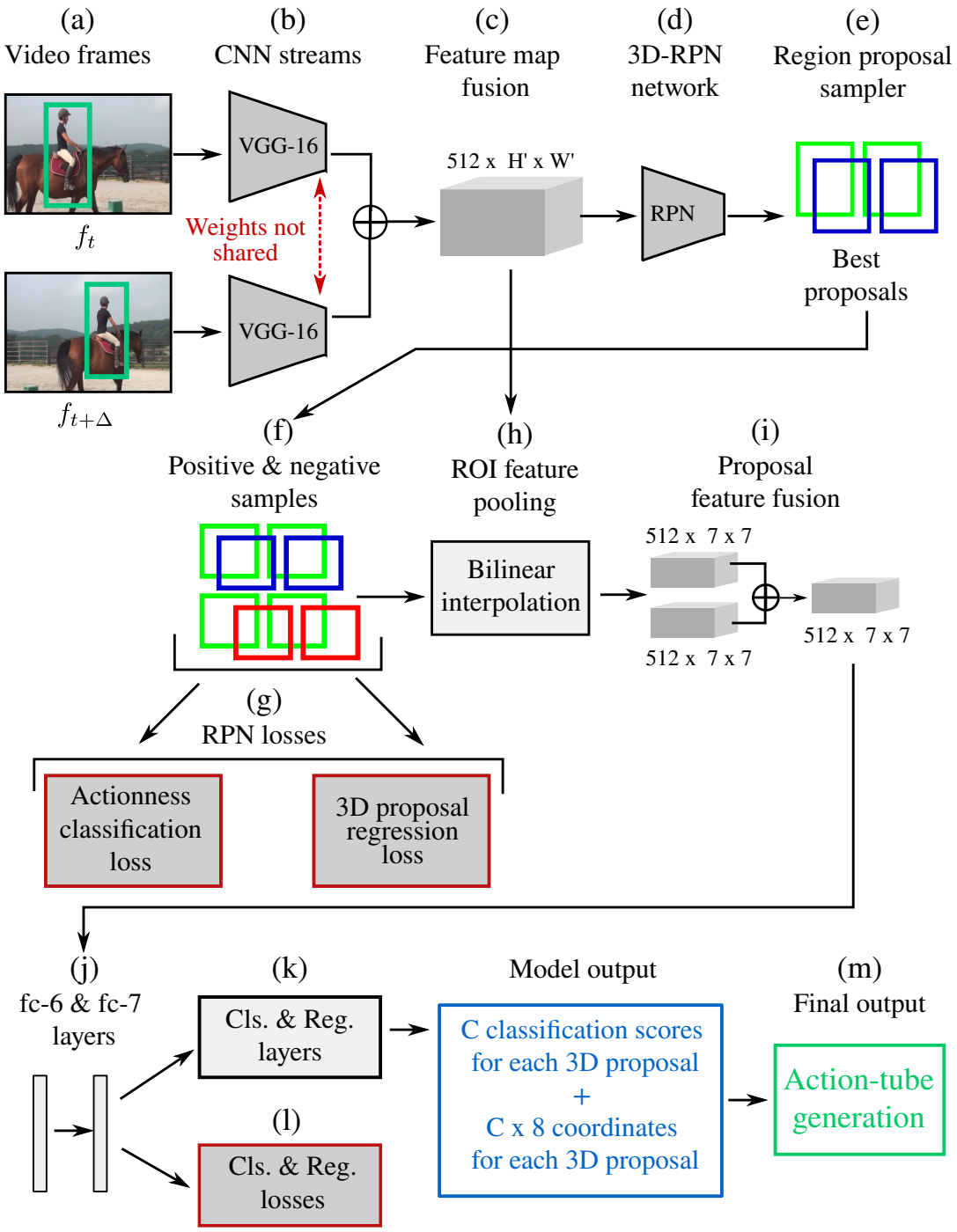


Figure 7.2: At train time, the input to the network is a pair of successive video frames ***(a)*** *which are processed through two parallel VGG-16 networks* ***(b)****. The feature maps generated by the last convolution layers are fused* ***(c)*** *and the fused feature map is fed to a 3D-RPN network* ***(d)****. The RPN generates 3D region proposals and their associated* actionness *[51] scores which are then sampled as positive and negative training examples* ***(f)*** *by a proposal sampler* ***(e)****. The sampled proposals and their scores are used to compute the* actionness *and 3D proposal regression losses* ***(g)****. Subsequently, a bilinear feature pooling* ***(h)*** *and an element-wise feature fusion* ***(i)*** *are used to obtain a fixed sized feature representation for each sampled 3D proposal. Finally, the pooled and fused features are passed through fully connected (FC6 & FC7)* ***(j)****, classification and regression* ***(k)*** *layers to train for action classification and a micro-tube regression. At test time, the predicted micro-tubes are linked in time by the action-tube generator* ***(m)****.*

7.2 Overview of the approach

Our proposed network architecture (Figure 7.2) employs and adapts some of the architectural components recently proposed in [29, 36]. At training time, the input to the model is a pair of successive video frames **(a)** which are fed to two parallel CNNs **(b)** (Section 7.3.1). The output feature maps of the two CNNs are fused **(c)** and passed as input to a 3D region proposal network (3D-RPN) **(d)** (Section 7.3.2). The 3D-RPN network generates 3D region proposals and their associated *actionness* [51] scores, which are then sampled as positive and negative training examples **(f)** by a proposal sampler **(e)** (Section 7.3.3). A training mini-batch of 256 examples are constructed from these positive and negative samples. The mini-batch is firstly used to compute the *actionness* classification and 3D proposal regression losses **(g)** (Section 7.4.1), and secondly, to pool CNN features (for each 3D proposal) using a bilinear interpolation layer **(h)** (Section 7.3.4).

In order to interface with the fully connected layers **(j)** (Section 7.3.5), bilinear interpolation is used to get a fixed-size feature representation for each variably sized 3D region proposal. As our 3D proposals consist of a pair of bounding boxes, we apply bilinear feature pooling independently on each bounding box in a pair, which gives rise to two fixed-size pooled feature maps of size $[512 \times kh \times kw]$, where $kh = kw = 7$ for each 3D proposal. We then apply element-wise fusion **(i)** (Section 7.3.4) to these 2 feature maps. Each pooled and then fused feature map (representing a 3D proposal) is passed to two fully connected layers (FC6 and FC7)) **(j)** (Section 7.3.5). The output of the FC7 layer is a fixed sized feature vector of shape $[4096 \times 1]$. These 4096 dimension feature vectors are then used by a classification and a regression layers **(k)** (Section 7.3.5) to output **(1)** $B \times C$ classification scores and **(2)** $B \times C \times 8$ coordinate values where B is the number of 3D proposals in a training mini-batch and C is the number of action categories in a given dataset.

At test time we select the top 1000 predicted micro-tubes by using non-maximum suppression, modified to work with pairs of bounding boxes and pass these to an action-tube generator **(m)** (Section 7.5) which links those micro-tubes in time. At both training and test time, our model receives as input successive video frames f_t, $f_{t+\Delta}$. At training time we generate training pairs using 2 different Δ values 1 and 2 (Section 7.6.1). At test time we fix $\Delta = 1$. As we show in the Section 7.7.4, even consecutive frames ($\Delta = 1$) carry significantly different information which affects the overall video-mAP. Throughout

this chapter, "3D region proposals" denotes the RPN-generated pairs of anchor boxes regressed by the middle layer (Figure 7.2 **(g)**), and later by the end layer (Figure 7.2 **(l)**). Whereas, "micro-tubes" refers to the pairs of either ground truth bounding-boxes or pairs of detection bounding-boxes predicted by the network at test time (Section 2.3 & Figure 2.4).

7.3 Network Architecture

All the stages of Figure 7.2 are described below in detail.

7.3.1 Convolutional Neural Network

The convolutional (conv) layers of our network follow the VGG-16 architecture [27]. We use two parallel VGG-16 networks (Figure 7.2 **(b)**) to apply convolution over a pair of successive video frames. The weights are not shared between these two CNNs, and thus, during training they learn two different sets of weights (Section 2.2.1). Each VGG-16 has 13 conv layers intermixed with 5 max pooling layers. Each conv layer has a 3×3 filter and 1×1 stride and padding. Each max pooling layer has filter shape 2×2. We discard all the VGG-16 layers after the last (13-th) conv layer.

Feature map fusion. Our network takes two successive video frames f_t and $f_{t+\Delta}$ as inputs. For a input video frame of shape $[3 \times H \times W]$, the last conv layer of each VGG-16 outputs a feature map of shape $[D \times H' \times W']$ where $D = 512$, $H' = \frac{H}{16}$, and $W' = \frac{W}{16}$. We fuse the two conv feature maps produced by the two parallel VGG-16 networks using element-wise sum fusion (Figure 7.2 **(c)**).
As a consequence, the fused feature map encodes both appearance *and* motion information (for frames f_t and $f_{t+\Delta}$), which we pass as input to our 3D-RPN network.

Our new 3D region proposal network (Figure 7.2 **(d)**) builds on the basic RPN structure [29] to propose a fully convolutional network which can generate 3D region proposals via a number of significant architectural changes.

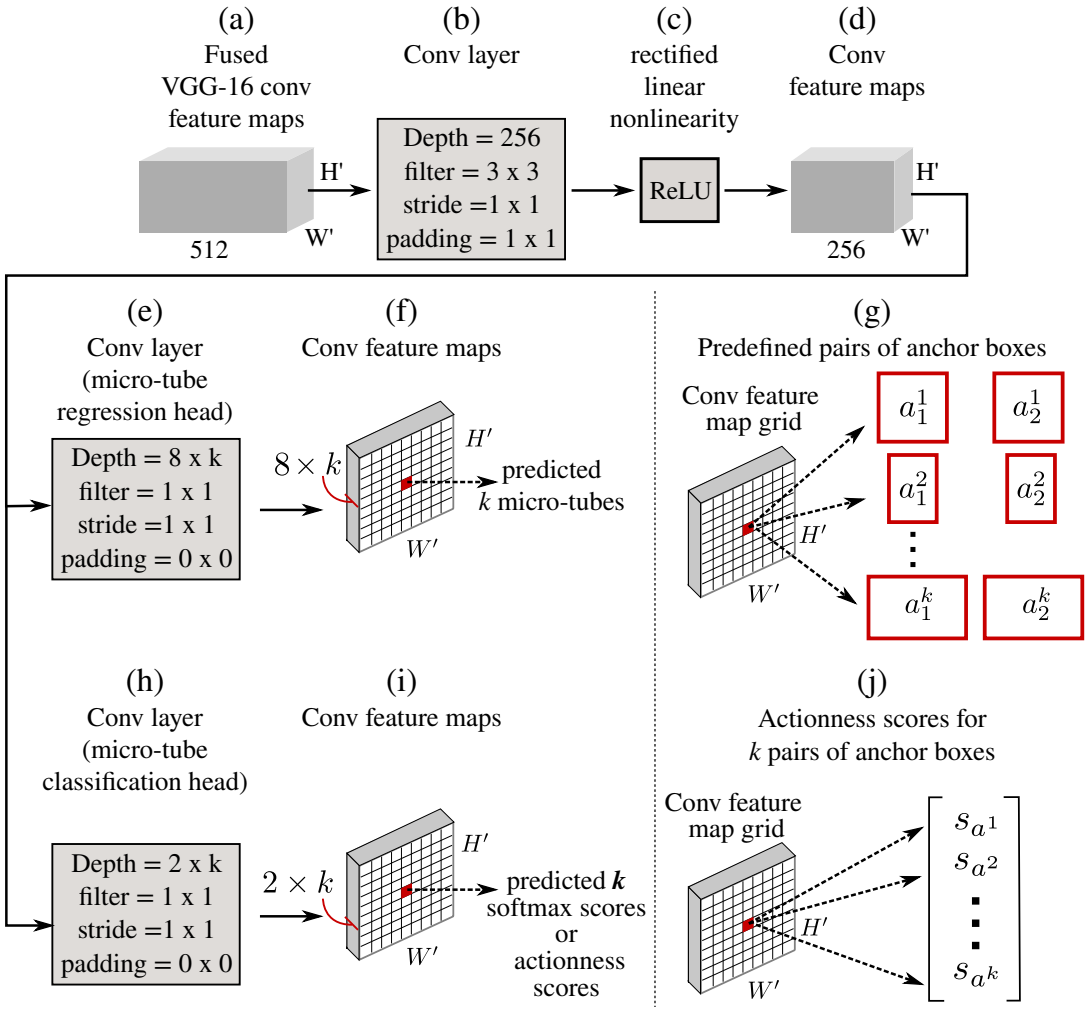


Figure 7.3: 3D-RPN architecture.

7.3.2 3D region proposal network

3D region proposal generation. As we explained, unlike a classical RPN [29] which generates region proposals (rectangular bounding boxes) per image, our 3D-RPN network generates (video) region proposals spanning a pair of video frames. A single proposal thus consists of a pair of rectangular bounding boxes. The input to our 3D-RPN is a fused VGG-16 feature map (Figure 7.2 **(c)**) of size $[512 \times H' \times W']$. We generate anchor boxes in a similar way as in [29]: namely, we project back each point in the $H' \times W'$ grid (of the input feature map) onto the original image plane of size $H \times W$. For each projected point we generate k pairs of anchor boxes of different aspect ratios.

Let $(x_{a_i}, y_{a_i}, w_{a_i}, h_{a_i})$ denote the centroid, width and height of the anchor boxes in a pair. We use the subscript i to index the two boxes in a pair, i.e. $i = \{1, 2\}$. Similarly, $(x_{g_i}, y_{g_i}, w_{g_i}, h_{g_i})$ refer to the centroid, width and height of the ground

truth pair. We can transform a pair of input anchor boxes into a predicted pair of ground truth boxes via[1]:

$$\begin{aligned} x_g &= x_a + \phi_x w_a & y_g &= y_a + \phi_y h_a \\ w_g &= w_a \exp(\phi_w) & h_g &= h_a \exp(\phi_h) \end{aligned} \tag{7.1}$$

where (ϕ_{x_i}, ϕ_{y_i}) specify a scale-invariant translation of the center of the anchor boxes, and (ϕ_{w_i}, ϕ_{h_i}) specify a log-space translation of their width and height.

Both RPN and the micro-tube regression layer (Figure 7.2 **(k)**) predict the bounding box regression offsets $(\phi_{x_i}, \phi_{y_i}, \phi_{w_i}, \phi_{h_i})$. Our anchor generation approach differs from that of [29], in the sense that we generate k *pairs* of anchors instead of k anchors.

Network architecture. The network architecture of our 3D-RPN is depicted in Figure 7.3. To encode the location information of each pair of anchors, we pass the fused VGG-16 feature map through a 3×3 convolution **(b)**, a rectified linear nonlinearity **(c)**, and two more 1×1 convolution (**(e)** and **(h)**) layers. The first conv layer **(b)** consists of 256 convolution filters with 1×1 stride and padding, resulting in a feature map of size $[256 \times H' \times W']$ **(d)**. The second conv layer **(e)** has $8 \times k$ convolution filters with 1×1 stride and does not have padding. It outputs a feature map of shape $[(8 \times k) \times H' \times W']$ **(f)** which encodes the location information (8 coordinate values) of $[k \times H' \times W']$ pairs of anchor boxes **(g)**. The third conv layer **(h)** is the same as **(e)**. The only difference is in the number of filters which is $2 \times k$ to encode the actionness score (i.e. probability of action or no-action) **(j)** for each k pairs of anchors.

As RPN is a fully convolutional neural network, classification and regression weights are learned directly from the convolution features, whereas in the fully connected layers (Section 7.3.5) we apply linear transformation layers for classification and regression. In our 3D-RPN, the convolution layer **(e)** is considered as the regression layer, as it outputs the 8 regression offsets per pair of anchor boxes; the convolution layer **(h)** is the classification layer.

7.3.3 3D region proposal sampling

Processing all the resulting region proposals is very expensive. For example, with $k = 12$ and a feature map of size $[512 \times 38 \times 50]$, we get $12 \times 38 \times 50 = 22800$ pairs of anchor boxes. For this reason, we subsample them during both training

[1]We removed the subscript i in Equation 7.1 for sake of simplicity.

and testing following the approach of [29] (Figure 7.2 **(e)**). We only make a slight modification in the sampling technique, as in our case one sample consists of a pair of bounding boxes, rather than a single box.

Training time sampling. During training, we compute the spatial IoU (Section 4.2.4) between a pair of ground truth boxes $\{G_t, G_{t+\Delta}\}$ and a pair of proposal boxes $\{P_1, P_2\}$, so that, $\psi_1 = IoU(G_t, P_1)$ and $\psi_2 = IoU(G_{t+\Delta}, P_2)$. We consider $\{P_1, P_2\}$ as a positive example if $\psi_1 \geq 0.5$ and $\psi_2 \geq 0.5$, that is both IoU values are above 0.5. When enforcing this condition, there might be cases in which we do not have any positive pairs. To avoid such cases, we also consider as positive pairs those which have maximal mean IoU $(\psi_1 + \psi_2)/2$ with the ground truth pair. As negative examples we consider pairs for which both IoU values are below 0.3.

We construct a minibatch of size B in which we can have at most $B_p = B/2$ positive and $B_N = B - B_P$ negative training samples. We set $B = 256$. Note that the ground truth boxes $\{G_t, G_{t+\Delta}\}$ in a pair belong to a same action instance but come from two different video frames $\{f_t, f_{t+\Delta}\}$. As there may be multiple action instances present, during sampling one needs to make sure that a pair of ground truth boxes belongs to the same instance. To this purpose, we use the ground truth tube-id provided in the datasets to keep track of instances.

Test time sampling. During testing, we use non-maximum suppression (NMS) to select the top $B = 1000$ proposal pairs. We made changes to the NMS algorithm to select the top B pairs of boxes based on their confidence. In NMS, one first selects the box with the highest confidence, to then compute the IoU between the selected box and the rest. In our modified version (i) we first select the pair of detection boxes with the highest confidence; (ii) we then compute the mean IoU between the selected pair and the remaining pairs, and finally (iii) remove from the detection list pairs whose IoU is above an overlap threshold th_{nms}.

7.3.4 Bilinear Interpolation

The sampled 3D region proposals are of different sizes and aspect ratios. We use *bilinear interpolation* [34, 35] to provide a fixed-size feature representation for them, necessary to pass the feature map of each 3D region proposal to the fully connected layer fc6 of VGG-16 (Figure 7.2 **(j)**), which indeed requires a fixed-size feature map as input. Whereas recent action detection methods [32, 33]

use max-pooling of region of interest (RoI) features which only backpropagates the gradients w.r.t. convolutional features, bilinear interpolation allows us to backpropagate gradients with respect to both (a) convolutional features and (b) 3D RoI coordinates. Further, whereas [32, 33] train appearance and motion streams independently, and perform fusion at test time, our model requires one-time training, and feature fusion is done at training time.

Feature fusion of 3D region proposals. As a 3D proposal consists of a pair of bounding boxes, we apply bilinear feature pooling independently to each bounding box in the pair. This yields two fixed-size pooled feature maps of size $[D \times kh \times kw]$ for each 3D proposal. We then apply element-wise sum fusion (Figure 7.2 **(i)**) to these 2 feature maps, producing an output feature map of size $[D \times kh \times kw]$. Each fused feature map encodes the appearance and motion information of (the portion of) an action instance which may be present within the corresponding 3D region proposal. In this work, we use $D = 512$, $kh = kw = 7$.

7.3.5 Fully connected layers

Our network employs two fully connected layers FC6 and FC7 (Figure 7.2 **(j)**), followed by an action classification layer and a micro-tube regression layer (Figure 7.2 **(k)**).
The fused feature maps (Section 7.3.4) for each 3D proposal are flattened into a vector and passed through FC6 and FC7. Both layers use rectified linear units and dropout regularisation [36]. For each 3D region proposal, the FC7 layer outputs a 4096 dimension feature vector which encodes the appearance and motion features associated with the pair of bounding boxes. Finally, these 4096-dimensional feature vectors are passed to the classification and regression layers. The latter output $[B \times C]$ softmax scores and $[B \times C \times 8]$ bounding box regression offsets (Section 7.3.2), respectively, for B predicted micro-tubes and C action classes.

7.4 Network training

7.4.1 Multi-task loss function

As can be observed in Figures 7.2 and 7.3, our network contains two distinct classification layers. The *mid classification layer* (Figure 7.3 **(h)**) predicts the

probability p^m of a 3D proposal containing an action, $p^m = (p_0^m, p_1^m)$ over two classes (action vs. no action). We denote the associated loss by L_{cls}^m. The *end classification layer* (Figure 7.2 **(k)**) outputs a discrete probability distribution (per 3D proposal), $p^e = (p_0^e, ..., p_C^e)$, over $C+1$ action categories. We denote the associated loss as L_{cls}^e.

In the same way, the network has a mid (Figure 7.3 **(e)**) and an end (Figure 7.2 **(k)**) regression layer – the associated losses are denoted by L_{loc}^m and L_{loc}^e, respectively. Both regression layers output a pair of bounding box offsets ϕ^m and ϕ^e (Equation 7.1). We adopt the parameterization of ϕ (Section 7.3.2) given in [28]. Now, each training 3D proposal is labelled with a ground truth action class c^e and a ground truth micro-tube (Section 2.3 , Figure 2.4 & Figure 7.1) regression target g^e. We can then use the multi-task loss [29]:

$$\begin{aligned} L(p^e, c^e, \phi^e, g^e, p^m, c^m, \phi^m, g^m) = \\ \lambda_{cls}^e L_{cls}^e(p^e, c^e) + \lambda_{loc}^e [c \geq 1] L_{loc}^e(\phi^e, g^e) + \\ \lambda_{cls}^m L_{cls}^m(p^m, c^m) + \lambda_{loc}^m [c = 1] L_{loc}^m(\phi^m, g^m) \end{aligned} \tag{7.2}$$

on each labelled 3D proposal to jointly train for (i) action classification (p^e), (ii) micro-tube regression (ϕ^e), (iii) actionness classification (p^m), and (iv) 3D proposal regression (ϕ^m). Here, $L_{cls}^e(p^e, c^e)$ and $L_{cls}^m(p^m, c^m)$ are the cross-entropy losses for the true classes c^e and c^m respectively, where c^m is 1 if the 3D proposal is positive and 0 if it is negative, and $c^e = \{1, ..., C\}$.

The second term $L_{loc}^e(\phi^e, g^e)$ is defined over an 8-dim tuple of ground truth micro-tube regression target coordinates:

$$g^e = \Big(\{g_{x_1}^e, g_{y_1}^e, g_{w_1}^e, g_{h_1}^e\}, \{g_{x_2}^e, g_{y_2}^e, g_{w_2}^e, g_{h_2}^e\}\Big)$$

and the corresponding predicted micro-tube tuple:

$$\phi^e = \Big(\{\phi_{x_1}^e, \phi_{y_1}^e, \phi_{w_1}^e, \phi_{h_1}^e\}, \{\phi_{x_2}^e, \phi_{y_2}^e, \phi_{w_2}^e, \phi_{h_2}^e\}\Big).$$

The fourth term $L_{loc}^m(\phi^m, g^m)$ is similarly defined over a tuple g^m of ground truth 3D proposal regression target coordinates and the associated predicted tuple ϕ^m. The Iverson bracket indicator function $[c \geq 1]$ in (7.2) returns 1 when $c^e \geq 1$ and 0 otherwise; $[c = 1]$ returns 1 when $c^m = 1$ and 0 otherwise.

For both regression layers we use a smooth $L1$ loss in transformed coordi-

nate space as suggested by [29]. The hyper-parameters λ^e_{cls}, λ^e_{loc}, λ^m_{cls} and λ^m_{loc}, in Equation 7.2 weigh the relative importance of the four loss terms. In the following we set to 1 all four hyper-parameters.

7.4.2 Optimisation

We follow the end-to-end training strategy of [36] to train the entire network in a single optimisation step. We use stochastic gradient descent (SGD) to update the weights of the two VGG-16 convolutional networks, with a momentum of 0.9. To update the weights of other layers of the network, we use the Adam [162] optimiser, with parameter values $\beta_1 = 0.9$, $\beta_2 = 0.99$ and a learning rate of 1×10^{-6}. During the 1st training epoch, we freeze the weights of the convolution networks and update only the weights of the rest of the network. We start fine-tuning the layers of the two parallel CNNs after completion of 1st epoch. The first four layers of both CNNs are not fine-tuned for sake of efficiency. The VGG-16 pretrained ImageNet weights are used to initialise the convolutional nets. The rest of the network's weights are initialised using a Gaussian with $\sigma = 0.01$.

7.5 Action-tube generation

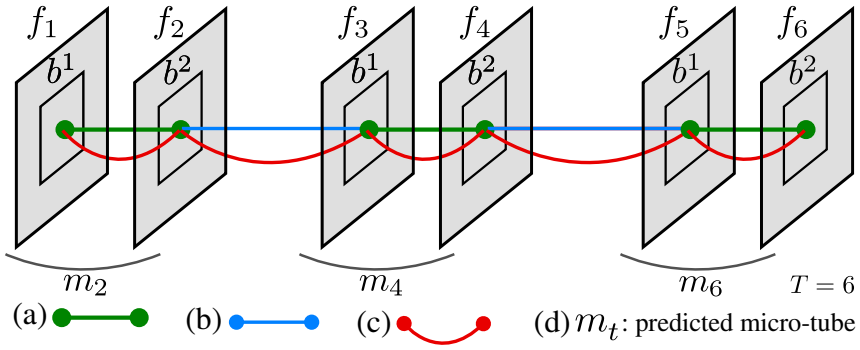


Figure 7.4: ***(a)*** *The temporal associations learned by our network;* ***(b)*** *Our micro-tube linking algorithm requires $(T/2 - 1)$ connections;* ***(c)*** *the $T - 1$ connections required by [33]'s approach.*

Once the predicted micro-tubes are regressed at test time, they need to be linked up to create complete action tubes associated with an action instance. To do this we introduce here a new action tube generation algorithm which is an evolution of that presented in [33]. There, temporally untrimmed action paths are first generated in a first pass of dynamic programming. In a second pass, paths are temporally trimmed to detect their start and end time. Here we modify the first

pass of [33] and build action paths using the temporal associations learned by our network. We use the second pass without any modification.

Linking up micro tubes (Figure 7.4) is not the same as linking up frame-level detections as in [33]. In the Viterbi forward pass of [33], the edge scores between bounding boxes belonging to consecutive video frames (i.e., frame f_t and f_{t+1}) are first computed. Subsequently, a DP (dynamic programming) matrix is constructed to keep track of the box indices with maximum edge scores. In the Viterbi backward pass, all consecutive pairs of frames, i.e, frames $\{1,2\},\{2,3\},\ldots$ are traversed to join detections in time. Our linking algorithm saves 50% of the computing time, by generating edge scores between micro-tubes (which only needs $T/2-1$ iterations, Figure 7.4) rather than between boxes from consecutive frames (which, in the forward pass, needs $T-1$ iterations). In the backward pass, the algorithm connects the micro-tubes as per the max edge scores.

Recall that a predicted micro-tube consists of a pair of bounding boxes (Figure 7.4), so that $m = \{b^1, b^2\}$. In the first pass action-specific paths $\mathbf{p}_c = \{m_t, t \in I = \{2,4,...,T-2\}\}$, spanning the entire video length are obtained by maximising via dynamic programming [14]:

$$E(\mathbf{p}_c) = \sum_{t \in I} s_c(m_t) + \lambda_o \sum_{t \in I} \psi_o\left(b^2{}_{m_t}, b^1{}_{m_{t+1}}\right), \tag{7.3}$$

where $s_c(m_t)$ denotes the softmax score (Section 7.3.5) of the predicted micro-tube m at time step t, the overlap potential $\psi_o(b^2{}_{m_t}, b^1{}_{m_{t+1}})$ is the IoU between the second detection box $b^2{}_{m_t}$ which forms micro-tube m_t and the first detection box $b^1{}_{m_{t+1}}$ of micro-tube m_{t+1}. Finally, λ_o is a scalar parameter weighting the relative importance of the pairwise term. By recursively removing the detection micro-tubes associated with the current optimal path and maximising (Section 7.3) for the remaining micro-tubes we can account for multiple co-occurring instances of the same action class.

7.6 Model evaluation

Please refer Section 4.2.3 for the frame-mAP and video-mAP evaluation metrics used in this section.

Table 7.1: Impact of different positive IoU thresholds on detection performance (video-mAP).

IoU threshold δ	0.1	0.2	0.3
Model-0-7	64.04	54.83	44.664
Model-0-5	**68.85**	**60.06**	**49.78**

7.6.1 Training data sampling strategy

As the input to our model is a pair of successive video frames and their associated ground truth micro-tubes, training data needs to be passed in a different way than in the frame-level training approach [14, 20, 32, 33], where inputs are individual video frames. In our experiments, we use 3 different sampling schemes to construct training examples using different combinations of successive video frames (Figure 7.1 **(b)**): (1) *scheme-11* generates training examples from the pairs of frames {t=1,t=2}, {t=2,t=3} ...; *scheme-21* uses the (non-overlapping) pairs {1,2}, {3,4} ...; *scheme-32* constructs training samples from the pairs {1,3}, {4,6} ... We first show how a proper positive IoU threshold is essential during the sampling of 3D region proposals at training time (Section 7.3.3). Secondly, we assess whether our proposed network architecture, coupled with the new data sampling strategies (Section 7.6.1), improves detection performance. We then show that our model outperforms the appearance-based model of [33]. Finally, we compare the performance of the overall detection framework with the state-of-the-art.

7.6.2 Impact of different positive IoU thresholds on detection performance

We train our model on UCF-101 using two positive IoU thresholds: 0.7 and 0.5 (Section 7.3.3). The detection results (video-mAP) of these two models (Model-0-7 & -0-5) are shown in Table 7.1. Whereas [29] recommends an IoU threshold of 0.7 to subsample positive region proposals during training, in our case we observe that an IoU threshold of 0.5 works better with our model. Indeed, during sampling we compute IoUs between pairs of bounding boxes and then take the mean IoU to subsample (Section 7.3.3). As the ground truth boxes (micro-tubes) are connected in time and span different frames, it is harder to get enough positive examples with a higher threshold like 0.7. Therefore, in the remainder we use an IoU of 0.5 for evaluation.

7.6.3 Impact of our training data sampling strategy on detection performance

JHMDB-21 frame-mAP. We first generate a J-HMDB-21 training set using the *scheme-11* (Section 7.6.1) and train our model. We then generate another training set using *scheme-32*, and train our model on the combined training set (*set-11+32*). Table 7.2 and Table 7.3 show per class frame-APs and video-APs obtained using these two models. We can observe that out of 21 JHMDB action classes, the frame-APs of 15 classes and video-APs of 12 classes actually improve when training the model on the new combined trainset (*set-11+32*). Overall performance increases by 1.64% for frame-AP and 1.04% for video-AP at $\delta = 0.5$ indicating that the network learns temporal association more efficiently when it is trained on pairs generated from different combinations of successive video frames.

JHMDB-21 video-mAP. The two above trained models are denoted by *Model-11* and *Model-11+32* in Table 7.5, where the video-mAPs at different IoU threshold for these two models are shown. Although the first training strategy *scheme-11* already makes use of all the video frames present in J-HMDB-21 training splits, when training our model using the combined trainset we observe an improvement in the video-mAP of 1.04% at $\delta = 0.5$.

7.6.4 Impact of exploiting appearance features

Further, we show that our model exploits appearance features (raw RGB frames) efficiently, contributing to an improvement of video-mAP by 3.5% over [33]. We generate a training set for UCF-101 split 1 using the training *scheme-21* and compare our model's performance with that of the appearance-based model (*A) of [33]. We show the comparison in Table 7.4.

Among the 24 UCF-101 action classes, our model exhibits better video-APs for 14 classes, with an overall gain of 3.5%. We can observe that, although trained on appearance features only, the proposed model improves the video-APs significantly for action classes which exhibit a large variability in appearance and motion. Note that, the proposed network learns to encode the motion patterns implicitly (during training) from pairs of successive RGB frames (i.e. f_t and $f_{t+\Delta}$). Also, our model achieves relatively better spatio-temporal detection on action classes associated with video sequences which are significantly temporally untrimmed, such as *basketball dunk*, *golf swing*, *diving* with relative video-AP

Table 7.2: Impact of our training data sampling strategy on per class frame-AP at IoU threshold $\delta = 0.5$, JHMDB-21 dataset (averaged over 3 splits).

frame-AP(%)	brushHair	catch	clap	climbStairs	golf	jump	kickBall	pick
ours (*)	46.4	40.7	31.9	62.3	91.0	4.3	17.3	29.5
ours (**)	43.7	43.6	33.0	61.5	91.8	5.6	23.8	31.5
Improvement	-2.6	**2.9**	**1.0**	-0.8	**0.7**	**1.2**	**6.4**	**1.9**
Gkioxari *et al.* [14]	65.2	18.3	38.1	39.0	79.4	7.3	9.4	25.2
Wang *et al.* [134]	60.1	34.2	56.4	38.9	83.1	10.8	24.5	38.5
Weinzaepfel *et al.* [20]	73.3	34.0	40.8	56.8	93.9	5.9	13.8	38.5
Peng *et al.* [32]	75.8	38.4	62.2	62.4	99.6	12.7	35.1	57.8

frame-AP(%)	pour	pullup	push	run	sBall*	sBow*	sGun*	sit
ours (*)	86.2	82.7	66.9	35.5	33.9	78.2	49.7	11.7
ours (**)	91.8	84.1	73.1	32.3	33.3	81.4	55.1	12.4
Improvement	**5.5**	**1.4**	**6.1**	-3.2	-0.6	**3.2**	**5.4**	**0.6**
Gkioxari *et al.* [14]	80.2	82.8	33.6	11.6	5.6	66.8	27.0	32.1
Wang *et al.* [134]	71.5	67.5	21.3	19.8	11.6	78.0	50.6	10.9
Weinzaepfel *et al.* [20]	88.1	89.4	60.5	21.1	23.9	85.6	37.8	34.9
Peng *et al.* [32]	96.8	97.3	79.6	38.1	52.8	90.8	62.7	33.6

frame-AP(%)	stand	sBBall*	throw	walk	wave	**mAP**	–	–
ours (*)	13.8	57.1	21.3	27.8	27.1	43.6	–	–
ours (**)	14.7	56.3	22.2	24.7	29.4	45.0	–	–
Improvement	**0.8**	-0.8	**0.8**	-3.1	**2.3**	**1.4**	–	–
Gkioxari *et al.* [14]	34.2	33.6	15.5	34.0	21.9	36.2	–	–
Wang *et al.* [134]	43.0	48.9	26.5	25.2	15.8	39.9	–	–
Weinzaepfel *et al.* [20]	49.2	36.7	16.8	40.5	20.5	45.8		
Peng *et al.* [32]	48.9	62.2	25.6	59.7	37.1	58.5		

{Gkioxari [14],Wang [134],Weinzaepfel [20],Peng [32]}*et al.* exploit both appearance and flow features, whereas, our model only exploit appearance features.

sBall* : shootBall, sBow* : shootBow, sGun* : shootGun, sBBall* : swingBaseball.
*Model-11, **Model-11+32.

improvements of 16.9%, 10.8% and 1.5% respectively. We report significant gains in absolute video-AP for action categories *soccer juggling*, *pole vault*, *rope climbing*, *basketball dunk*, *ice dancing*, *golf swing* and *long jump* of 45.6%, 22.1%, 19.4%, 16.9%, 11.2% 10.8% and 8.3%, respectively.

7.6.5 Detection performance comparison with the state-of-the-art

Table 7.6 reports action detection results, averaged over the three splits of *J-HMDB-21*, and compares them with those to our closest competitors. Note that, although our model only trained using the appearance features (RGB images), it outperforms [14] which was trained using both appearance and optical flow

Table 7.3: Impact of our training data sampling strategy on per class video-AP at IoU threshold $\delta = 0.5$, JHMDB-21 dataset (averaged over 3 splits).

video-AP(%)	brushHair	catch	clap	climbStairs	golf	jump	kickBall	pick
ours (*)	53.9	54.4	39.8	68.2	96.1	5.69	39.6	34.9
ours (**)	51.9	54.5	41.2	66.6	94.8	7.8	48.7	33.7
Improvement	-1.9	**0.01**	**1.4**	-1.6	-1.2	**2.1**	**9.1**	-1.2
Gkioxari *et al.* [14]	79.1	33.4	53.9	60.3	99.3	18.4	26.2	42.0
Wang *et al.* [134]	76.4	49.7	80.3	43.0	92.5	24.2	57.7	70.5

video-AP(%)	pour	pullup	push	run	sBall*	sBow*	sGun*	sit
ours (*)	97.1	93.5	84.1	53.7	43.6	93.2	64.5	20.9
ours (**)	97.6	92.5	87.6	49.0	37.4	92.7	75.8	21.6
Improvement	**0.4**	-1.0	**3.4**	-4.7	-6.2	-0.5	**11.2**	**0.6**
Gkioxari *et al.* [14]	92.8	98.1	29.6	24.6	13.7	92.9	42.3	67.2
Wang *et al.* [134]	78.7	77.2	31.7	35.7	27.0	88.8	76.9	29.8

video-AP(%)	stand	sBBall*	throw	walk	wave	**mAP**	–	–
ours (*)	22.8	72.1	23.2	39.4	37.8	54.27		
ours (**)	27.1	73.3	24.3	37.7	44.7	55.31		
Improvement	**4.2**	**1.1**	**1.1**	-1.6	**6.8**	**1.04**		
Gkioxari *et al.* [14]	57.6	66.5	27.9	58.9	35.8	53.3		
Wang *et al.* [134]	68.6	72.8	31.5	44.4	26.2	56.4		

{Gkioxari [14],Wang [134]}*et al.* exploit both appearance and flow features, whereas, our model only exploit appearance features.

sBall* : shootBall, sBow* : shootBow, sGun* : shootGun, sBBall* : swingBaseball.
*Model-11, **Model-11+32.

Table 7.4: Per class video-AP comparison at IoU threshold $\delta = 0.2$, UCF-101-24 dataset.

video-AP(%)	BaDu	Bi	Di	Fe	FlGy	GoSw	IcDa	LoJu
Saha *et al.* [33] (*A)	22.7	56.1	89.7	86.9	93.8	59.9	59.2	41.5
ours	39.6	59.5	91.2	88.5	94.1	70.7	70.4	49.8
Improvement	**16.9**	**3.4**	**1.5**	**1.6**	**0.3**	**10.8**	**11.2**	**8.3**

video-AP(%)	PoVa	RoCl	Sk	SkJe	SoJu	WaDo	mAP	–
Saha *et al.* [33] (*A)	48.9	77.8	68.4	88	34.6	73.0	56.55	–
ours	71.0	97.2	74.0	92.9	80.2	73.6	60.06	–
Improvement	**22.1**	**19.4**	**5.6**	**4.9**	**45.6**	**0.6**	**3.5**	–

BaDu : BasketballDunk, Bi : Biking, Di : Diving, Fe : Fencing, FlGy : FloorGymnastics, GoSw : GolfSwing, IcDa : IceDancing, LoJu : LongJump, PoVa : PoleVault, RoCl : RopeClimbing, Sk : Skiing, SkJe : Skijet, SoJu : SoccerJuggling, WaDo : WalkingWithDog.
*A: appearance model.

Table 7.5: Impact of our training data sampling strategy on video-mAP, JHMDB-21 (averaged over 3 splits).

IoU threshold δ	0.1	0.2	0.3	0.4	0.5
Model-11	57.73	57.70	57.60	**56.81**	54.27
Model-11+32	**57.79**	**57.76**	**57.68**	56.79	**55.31**

Table 7.6: Spatio-temporal action detection performance (video-mAP) comparison with the state-of-the-art on J-HMDB-21.

IoU threshold δ	0.1	0.2	0.3	0.4	0.5
Gkioxari and Malik [14]	–	–	–	–	53.30
Wang *et al.* [134]	–	–	–	–	56.40
Weinzaepfel *et al.* [20]	–	63.1	–	–	60.70
Saha *et al.* [33] (Spatial Model)	52.99	52.94	52.57	52.22	51.34
Peng and Schmid [32]	–	**74.3**	–	–	**73.1**
Ours	57.79	57.76	57.68	56.79	55.31

Table 7.7: Spatio-temporal action detection performance (video-mAP) comparison with the state-of-the-art on UCF-101.

IoU threshold δ	0.1	0.2	0.3	0.5	0.75	0.5:0.95
Yu *et al.* [26]	42.8	26.50	14.6	–	–	–
Weinzaepfel *et al.* [20]	51.7	46.8	37.8	–	–	–
Peng and Schmid [32]	**77.31**	**72.86**	**65.70**	30.87	**01.01**	07.11
Saha *et al.* [33] (*A)	65.45	56.55	48.52	–	–	–
Saha *et al.* [33] (full)	76.12	66.36	54.93	–	–	–
Ours – ML	68.85	60.06	49.78	–	–	–
Ours – $ML - (*)$	70.71	61.36	50.44	32.01	0.4	9.68
Ours – $2PDP - (*)$	71.3	63.06	51.57	**33.06**	0.52	**10.72**

(*) cross validated alphas as in [33]; 2PDP - tube generation algorithm [33]
ML - our micro-tube linking algorithm.

features. Also, our model outperforms [33]'s spatial detection network.

Table 7.7 compares the action detection performance of our model on the UCF-101 dataset to that of current state of the art approaches. We can observe that our model outperforms [20, 26, 33] by a large margin. In particular, our appearance-based model outperforms [20] which exploits both appearance and flow features. Also notice, our method works better than that of [32] at higher IoU threshold, which is more useful in real-world applications. In Chapter 8, we show that when motion stream is added to the AMTnet model (i.e. AMTnet-Flow), it outperforms [32] (at majority of the IoU thresholds). In Section 8.5.5, we also discuss the upside of AMTnet-Flow over [32]'s approach.

7.7 Supporting experiments and discussion

7.7.1 Impact of the number of predicted 3D proposals

To investigate the effect of the number of predicted 3D proposals on detection performance, we generate video-mAPs using two different sets of detections on J-HMDB-21 dataset. One detection set is generated by selecting top 1000 3D proposals and another set is by selecting top 300 3D proposals at test time using NMS. Once the two sets of detections are extracted, predicted micro-tubes are then linked up in time to generate final action tubes. Subsequently, video-mAPs are computed for each set of action tubes. The corresponding video-mAPs for each detection set at different IoU thresholds are reported in Table 7.8. We denote these two detection sets in Table 7.8 as *Detection-1000* and *Detection-300*. It is quite apparent that reduced number of RPN proposals does not effect the detection performance.

Table 7.8: Impact of the number of predicted 3D proposals on video-mAP for J-HMDB-21 dataset (averaged over 3 splits).

IoU threshold δ	0.1	0.2	0.3	0.4	0.5
Detection-1000	57.79	57.76	57.68	56.79	**55.31**
Detection-300	**57.91**	**57.89**	**57.84**	**56.87**	55.26

7.7.2 Loss function hyper-parameters

We have four hyper-parameters λ^e_{cls}, λ^e_{loc}, λ^m_{cls} and λ^m_{loc}, in our multi-task loss function (Equation 7.2) which weigh the relative importance of the four loss terms. To investigate the effect of these hyper-parameters on video-mAP, we train our model with different combinations of these four hyper-parameters on J-HMDB-21 split-1. The trainset is generated as per *scheme-11* (Section 7.6.1). The video-mAPs of these trained models are presented in Table 7.9. We can observe that when the weights for the *mid classification* (λ^m_{cls}) and *regression* (λ^m_{loc}) layers' loss terms are too low (e.g. 0.1 & 0.05), the model has the worst detection performance. When all weights are set to 1, then the model exhibits good detection performance. However, we get the best video-mAPs with $\lambda^e_{cls} = 1.0$, $\lambda^e_{loc} = 1.0$, $\lambda^m_{cls} = 0.5$ and $\lambda^m_{loc} = 0.5$. In all our experiments we set all 4 weights to 1. As a future work, we will explore the setting [1.0, 1.0, 0.5, 0.5].

Table 7.9: Impact of different combinations of hyper-parameters on video-mAP for J-HMDB-21 split-1 train set.

Hyper-parameters				IoU threshold δ				
λ^e_{cls}	λ^e_{loc}	λ^m_{cls}	λ^m_{loc}	0.1	0.2	0.3	0.4	0.5
1.0	1.0	0.1	0.05	55.03	55.03	54.63	53.17	50.33
1.0	1.0	0.1	0.1	55.62	55.62	55.47	54.47	50.51
1.0	1.0	0.5	0.25	56.3	56.3	55.91	54.76	52.30
1.0	1.0	0.5	0.5	**57.3**	**57.13**	**56.79**	**55.82**	**53.81**
1.0	1.0	1.0	1.0	56.86	56.85	56.57	55.89	52.78

Model-11-2PDP

Table 7.10: An ablation study on J-HMDB-21 (split-01). Video-mAP is computed at IoU threshold $\delta = 0.5$.

Model	video-mAP (%)
Model-01	48.9
Model-02	52.7
Model-03	**57.1**

Model-01: Training pairs with identical frames
Model-02: Training pairs with consecutive frames (model-11)
Model-03: Training pairs with mixture of consecutive and successive frames (model-11+32)

7.7.3 Computing time required for training/testing

Computing time required for training. Saha *et al.*reported [163] that the state-of-the-art [14,20] action detection methods require at least 6+ days to train all the components (including fine-tuning CNNs, CNN feature extraction, one vs rest SVMs) of their detection pipeline for UCF-101 trainset (split-01). In our case, we need to train the model once which requires 96 hours for UCF-101 and 36 hours for J-HMDB-21 to train. The training and test time calculations are done considering a single NVIDIA Titan X GPU. The computing time requirement for different detection methods are presented in Table 7.11. Our model requires 2 days less training time as compared to [14, 20] on UCF-101 trainset.

Computing time required for testing. We compare video-level computing time required (during test time) of our method with [14, 20, 33] on J-HMDB-21 dataset. Note that our method takes the least computing time of 8.5 Sec./video as compared to [14, 20, 33] (Table 7.11).

Table 7.11: Computing time comparison for training and testing.

Methods	days (*)	Sec/video (**)
Gkioxari *et al.* [14]	6+	113.52
Weinzaepfel *et al.* [20]	6+	52.23
Saha *et al.* [33]	**3+**	10.89
ours	4	**8.5**

(*) Training time on UCF-101 dataset.
(**) Average detection time on J-HMDB-21.

7.7.4 Ablation study

Our model is not limited to learn from pairs of consecutive frames, but can learn from pairs at any arbitrary interval Δ (see Figure 7.1 **(a)**). To justify this claim, we conducted an ablation study of our model which is discussed below. For consecutive frames, we trained our model on J-HMDB-21 (split-01) dataset by passing training pairs composed of identical frames, e.g. passing the video frame pair $(65, 65)$ instead of $(65, 66)$. As you can see in Table 7.10, video-mAP drops significantly by 8.13% (at IoU threshold $\delta = 0.5$) which implies that the two streams do not output identical representations.

To double-check, we also extracted the two VGG-16 conv feature maps (see Figure 7.2 **(b)**) for each test frame pair $((f_t,\ f_{t+1}))$ of J-HMDB-21 and UCF-101 datasets. For each pair of conv feature maps, we first flattened them into feature vectors, and then computed the normalised $L2$ distance between them. For identical frames we found that the $L2$ distance is 0 for both J-HMDB-21 and UCF-101 datasets. Whereas, for consecutive frames it is quite high, in case of J-HMDB-21 the mean $L2$ distance is 0.67; for UCF-101 the mean $L2$ distance is 0.77 which again implies that the two streams generate significantly different feature encoding even for pairs consist of consecutive video frames.

Besides, as a part of the ablation study, per class frame- and video-APs of J-HMDB-21 dataset are reported in Table 7.2 and Table 7.3, and per class video-APs of UCF-101 are presented in Table 7.4.

7.7.5 Qualitative results

Spatio-temporal action detection results on UCF-101. We show the spatio-temporal action detection qualitative results in Figures 7.5 and 7.6. To demonstrate the robustness of the proposed detector against temporal action detection, we select those action categories which have highly temporally untrimmed videos. We select action classes *volleyball spiking*, *basketball dunk*

and *cricket bowling*. For *volleyball spiking* class, the average temporal extent of the action in each video is 40%, that means, the remaining 60% of the video does not contain any action. Similarly, for *basketball dunk* and *cricket bowling* classes, we have average durations 41% and 46% respectively.

Video clip **(a)** (Figures 7.5) has duration 107 frames and the action *volleyball spiking* takes place only between frames 58 to 107. Note that our method able to successfully detect the temporal extent of the action (alongside spatial locations) which closely matches the ground truth. We can observe similar quality of detection results for video clip **(b)** and **(c)** (Figures 7.5) which have durations 41 and 94 frames and the temporal extent of action instances are between frames 17 to 41 and frames 75 to 94 respectively for *basketball dunk* and *cricket bowling*. Video clips **(a)** and **(b)** in Figures 7.6 show some more spatio-temporal detection results for action classes *basketball dunk* and *cricket bowling*.

Figures 7.7 shows sample detection results on UCF-101. Note that in **(1)**, the 2nd "biker" is detected in spite of partial occlusion. Figures 7.7 **(1)**, **(2)**, **(3)** and **(5)** are examples of multiple action instance detection with complex real world scenarios like 3 fencers (Figures 7.7 **(2)**) and 3 bikers (Figures 7.7 **(3)**). Further, note that the detector is robust against *scale changes* as the 3rd fencer (Figures 7.7 **(2)**) and the 3rd biker (Figures 7.7 **(3)**) are detected accurately in spite of their relatively smaller shapes.

Spatio-temporal action detection results on J-HMDB-21. Figure 7.8 presents the detection results of our model on J-HMDB-21 dataset. In Figure 7.8 **(1)**, **(2)** and **(3)**, the actions "run" and "sit" are detected accurately in spite of large variations in illumination conditions, which shows that our detector is robust against *illumination changes*. In Figure 7.8 **(5)**, **(6)** and **(7)**, the actions "jump" and "run" are detected successfully. Note that due to fast motion, these video frames are affected by *motion blur*. Further, in Figure 7.8 **(9)** to **(12)**, actions "stand" and "sit" are detected with correct action labels. Even for human, it is hard to infer which instance belong to "stand" and "sit" class. This again tells that our classifier is robust against inter-class similarity.

7.8 Implementation details

We implement our method using Torch 7 [164]. To develop our codebase, we take coding reference from the publicly available repository [165]. We use the coding implementation of bilinear interpolation [166] (Section 7.3.4) for ROI feature

pooling. Our micro-tube linking algorithm (ML) (Section 7.5) is implemented in MATLAB.

In all our experiments, at training time we pick the top 2000 RPN generated 3D proposals using NMS (non-maximum suppression). At test time we select the top 1000 3D proposals. However, a lower number of proposals, e.g. top 300 proposals does not effect the detection performance, and increases the test time detection speed significantly. In Section 7.7.1, we show that extracting fewer 3D proposals (at test time) does not effect the detection performance. Shaoqing *et al.* [29] observed the same with Faster-RCNN.

For UCF-101, we report test time detection results (video-mAP) using two different action-tube generation algorithms. Firstly, we link the micro-tubes predicted by the proposed model (at test time) using our **m**icro-tube **l**inking (ML) algorithm (Section 7.5). we denote this as *"Ours-ML"* in Table 7.7. Secondly, we construct final action-tubes from the predicted micro-tubes using the **2 p**ass **d**ynamic **p**rogramming (2PDP) algorithm proposed by [33]. We denote this as *"Ours-2PDP"* in Table 7.7. The results in Table 7.1, 7.4, 7.5 and 7.6 are generated using our new micro-tube linking algorithm (*"Ours-ML"*). Further, we cross-validate the class-specific α_c as in Section 3.4 of [33], and generate action-tubes using these cross-validated α_c values. We denote the respective results using an asterisk (**'*'**) symbol in Table 7.7.

7.8.1 Mini-batch sampling

In a similar fashion [49], we construct our gradient descent mini-batches by first sampling N pairs of successive video frames, and then sampling R 3D proposals for each pair. In practice, we set $N = 1$ and $R = 256$ in all our experiments. We had one concern over this way of sampling training examples because, all the positive 3D proposals from a single training batch (i.e. a pair of video frames) belong to only one action category [2] (that is, they are correlated), which may cause slow training convergence. However, we experience a fast training convergence and good detection results with the above sampling strategy.

[2]Each video clip of UCF-101 and J-HMDB-21 is associated with a single class label. Therefore, a pair of video frames belongs to a single action class.

7.8.2 Data preprocessing

The dimension of each video frame in both J-HMDB-21 and UCF-101 is $[320 \times 240]$. We scale up each frame to dimension $[800 \times 600]$ as in [29]. Then we swap the RGB channels to BGR and subtract the VGG image mean $\{103.939, 116.779, 123.68\}$ from each BGR pixel value.

7.8.3 Data augmentation

We augment the training sets by flipping each video frame horizontally with a probability of 0.5.

7.8.4 Training batch

Our training data loader script constructs a training batch which consists of: a) a tensor of size $[2 \times D \times H \times W]$ containing the raw RGB pixel data for a pair of video frames, where $D = 3$ refers to the 3 channel RGB data, $H = 600$ is the image height and $W = 800$ is the image width; b) a tensor of size $[2 \times T \times 6]$ which contains the ground truth micro-tube annotation in the following format: [*fno tid* x_c y_c w h], where T is the number of micro-tubes, *fno* is the frame number of the video frame, *tid* is an unique identification number assigned to each individual action tube within a video, $\{x_c, y_c\}$ is the center and w and h are the width and height of the ground truth bounding box; c) a $[1 \times T]$ tensor storing the action class label for each micro-tube. The J-HMDB-21 (Model-11+32) train set has 58k training batches, and UCF-101 train set consists of 340k training batches.

7.8.5 Training iteration

Our model requires at least 2 training epochs because, in the first training epoch we freeze the weights of all the convolutional layers and only update the weights of the rest of the network. We start updating the weights of the convolutional layers (alongside other layers) in the second epoch. We stop the training after 195k and 840k iterations for J-HMDB-21 and UCF-101 respectively. The training times required for J-HMDB-21 and UCF-101 are 36 and 96 GPU hours respectively using a single GPU. The training time can be further reduced by using two or more GPUs in parallel.

7.8.6 Fusion methods

A fusion function $f : \vec{x}^t, \vec{x}^{t+\Delta}, \rightarrow y$ fuses two convolution feature maps $\vec{x}^t, \vec{x}^{t+\Delta} \in \mathbb{R}^{H' \times W' \times D}$ to produce an output map $y \in \mathbb{R}^{H' \times W' \times D}$, where W', H' and D are the width, height and number of channels of the respective feature maps [52]. In this work we experiment with the following fusion method.

Sum fusion. Sum fusion $y^{sum} = f^{sum}(\vec{x}^t, \vec{x}^{t+\Delta})$ computes the sum of the two feature maps at the same spatial locations, (i, j) and feature channels d:

$$y^{sum}_{i,j,d} = \vec{x}^t_{i,j,d} + \vec{x}^{t+\Delta}_{i,j,d} \tag{7.4}$$

where $1 \leq i \leq H', 1 \leq j \leq W', 1 \leq d \leq D$ and $\vec{x}^t, \vec{x}^{t+\Delta}, y \in \mathbb{R}^{H' \times W' \times D}$.

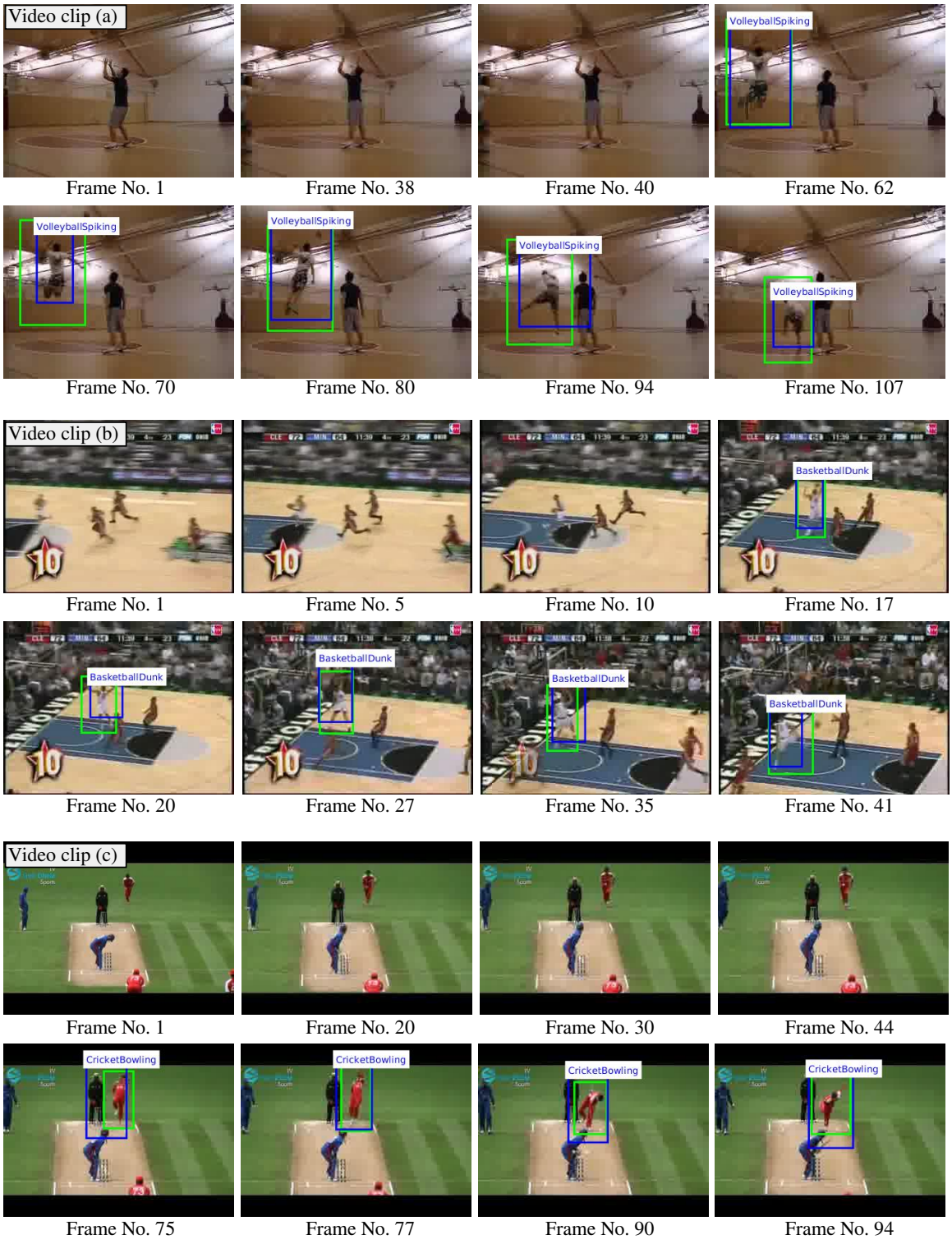


Figure 7.5: Spatio-temporal action detection results. Video clips ***(a)****,* ***(b)*** *and* ***(c)*** *are test videos belong to UCF-101 action classes "volleyball spiking", "basketball dunk" and "cricket bowling" respectively.*

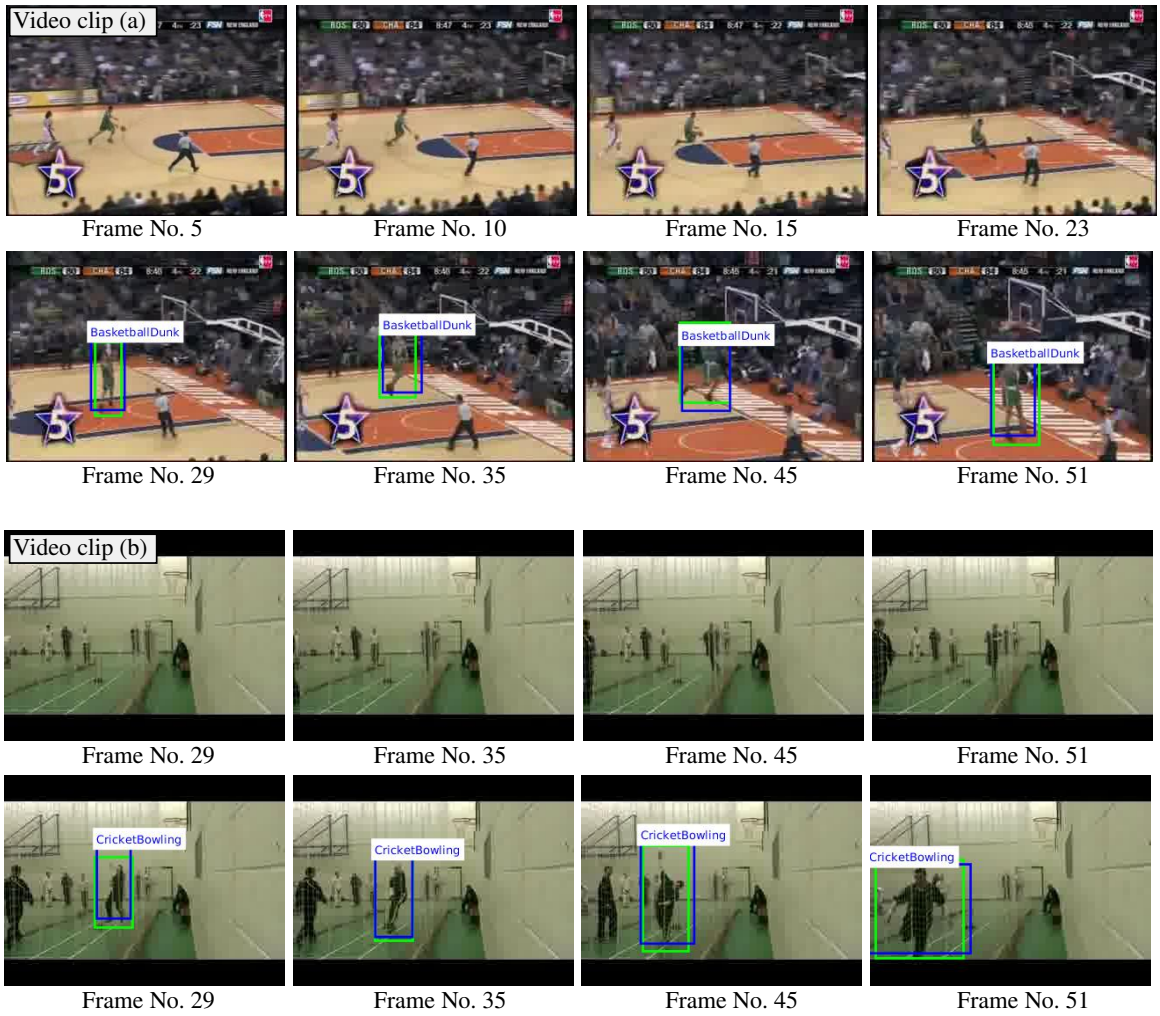


Figure 7.6: Spatio-temporal action detection results. Video clips ***(a)*** *and* ***(b)*** *are test videos belong to UCF-101 action classes "basketball dunk" and "cricket bowling" respectively.*

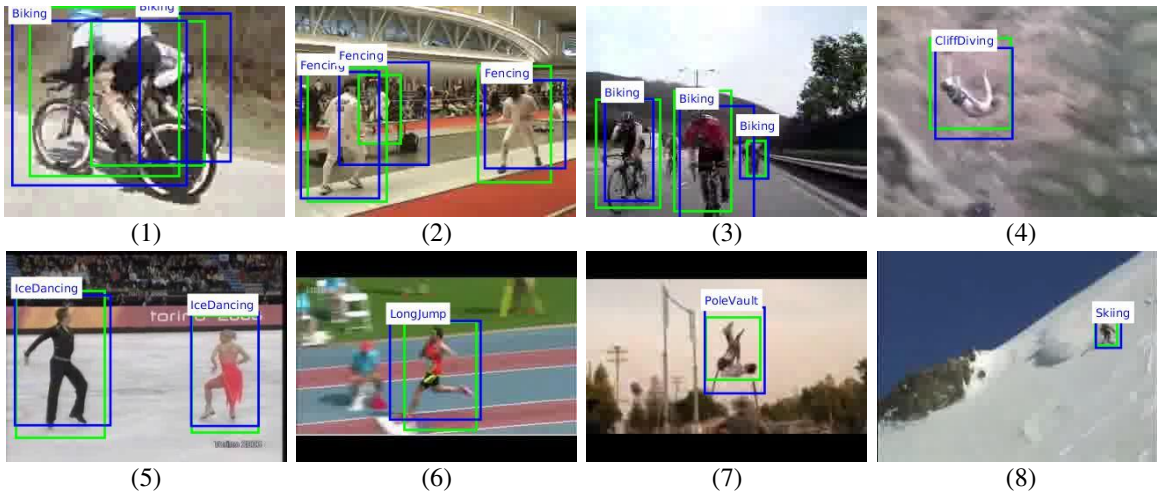


Figure 7.7: Additional sample detection results on UCF-101-24 dataset.

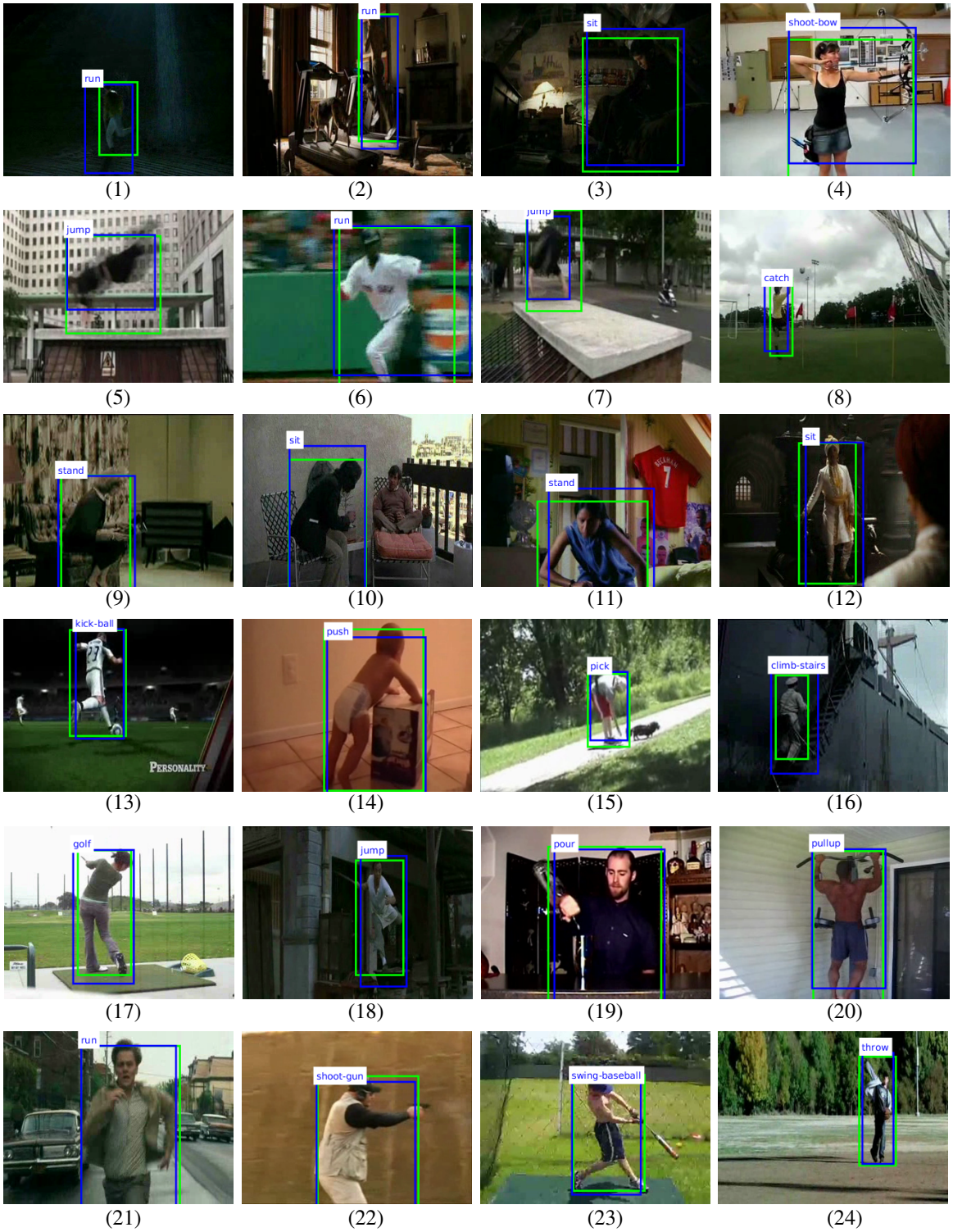


Figure 7.8: Spatio-temporal action detection results on J-HMDB-21 dataset.

Chapter 8

AMTnet-Flow: Improving Action-Micro-Tube Detection with Flow Stream

8.1 Introduction

Following from the previous chapter (Chapter 7), here we extend the AMTnet deep architecture to improve both action detection performance and speed. The main contributions of this chapter is as follows:

- We significantly improve the action detection performance of AMTnet by incorporating optical flow features (Section 8.5.1). We name this new deep network as *"AMTnet-Flow"* (Section 8.2).
- Unlike AMTnet, we train AMTnet-Flow on both nearby and widely separated frame pairs (i.e. pairs having short and long inter-frame distance), and at the test time, we improve the detection speed significantly by extracting action micro-tubes on long distance pairs (Section 8.4). At test time to extract longer detection micro-tubes, we propose an efficient *"box interpolation algorithm"* (Section 8.3.1) which populates the missing bounding boxes for intermediate frames by using linear interpolation of box coordinates. For temporal linking of longer micro-tubes (Section 8.3.2), we modify the micro-tube linking algorithm (Section 7.5) which further speeds up the tube generation step (Section 8.5.3).
- Unlike the test time fusion approach [14, 20, 32, 33], our new train time CNN feature fusion scheme allows the network to learn the pixel-wise correspondences between appearance and motion features [52].

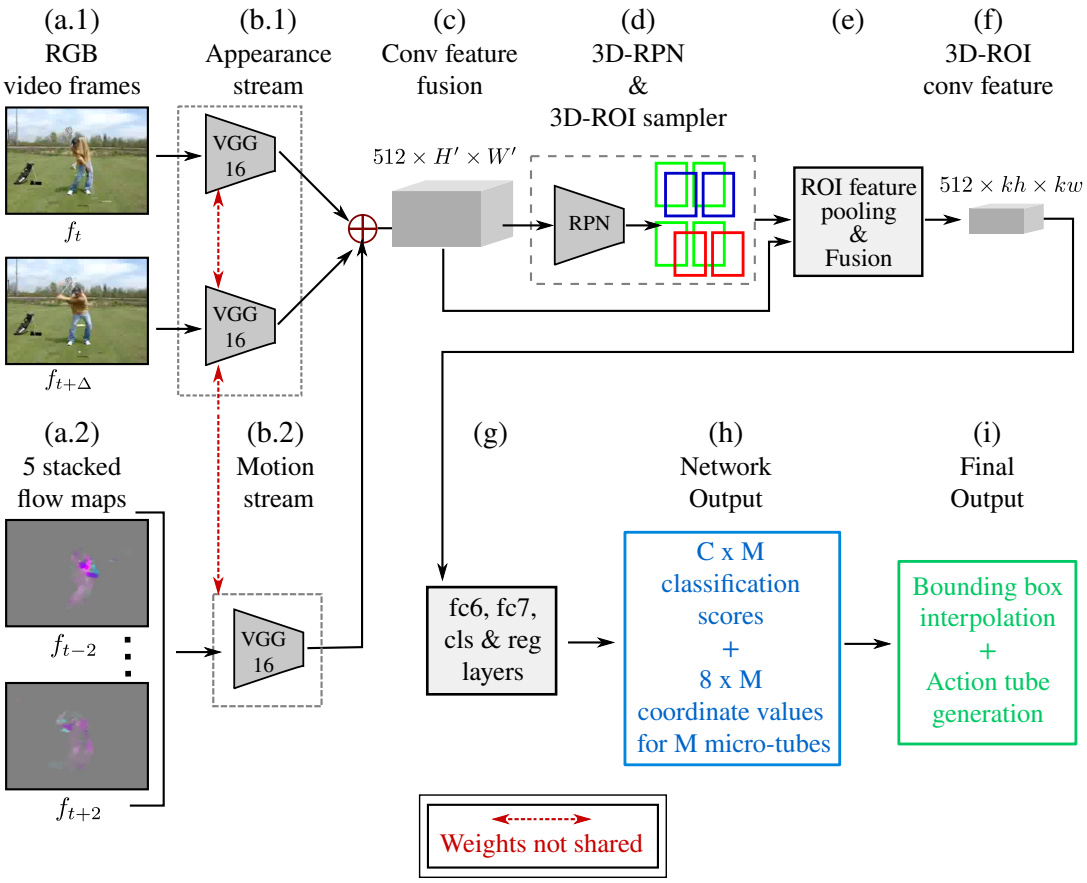


Figure 8.1: Overview of the proposed AMTnet-Flow action detection network.

Outline. This chapter is organized as follows. We first present the network architecture of AMTnet-Flow model in Sections 8.2. We then describe the bounding-box interpolation and action tube generation algorithms in Section 8.3. Section 8.4 discusses the train and test time data sampling strategies. Next, we present an extensive evaluation of the AMTnet-Flow model in Section 8.5. Finally, implementation details are provided in Section 8.6.

8.2 Network architecture

The network architecture used in this chapter is similar to AMTnet architecture (Chapter 7). We extend AMTnet by adding an additional motion stream alongside the existing two RGB flow streams. We use VGG-16 network architecture [27] for our two RGB and one flow stream (Figure 8.1 (b.1) & (b.2)). We keep all the layers of VGG-16 from the first layer to the last convolutional layer, and discard the remaining layers. In this chapter we call this modified VGG as VGG-16*.

8.2.1 Appearance streams

We use two parallel VGG-16* for our appearance streams (Figure 8.1 (b.1)). At each forward pass (during training and testing), these two CNNs receive two successive RGB video frames f_t and $f_{t+\Delta}$ (Figure 8.1 (a.1)) and process them through 13 convolutional layers (intermixed with 5 max pooling layers) in parallel. The output of each CNN is a convolutional feature map produced by the last conv layer which has dimension $[D \times H' \times W']$ where $D = 512$, $H' = \frac{H}{16}$, and $W' = \frac{W}{16}$, H is the input frame height, W is the frame width.

8.2.2 Motion stream

We use a single VGG-16* for the motion stream (Figure 8.1 (b.2)). The motion stream takes 5 stacked optical flow maps (Figure 8.1 (a.2)) [32] as input. We refer the readers to Section 8.6.4 and A.1 for motion stream implementation details and optical flow maps computation.

It processes the stacked flow maps in parallel to the appearance stream, and outputs a feature map of dimension $[D \times H' \times W']$. The 3 feature-maps are then fused using a sum fusion [52] and the resultant feature-map (Figure 8.1 (c)) is passed as input to a 3D-RPN.

Please refer Section 7.3.2 and 7.3.3 for 3D-RPN and 3D-RoI sampler respectively (Figure 8.1 (d)), Section 7.3.4 for RoI feature pooling (using bilinear interpolation) and fusion (Figure 8.1 (e)).

8.2.3 Fully connected, classification and regression layers

The features (Figure 8.1 (f)) of each positive and negative 3D-ROIs are processed through two fully connected (fc6 and fc7), a classification and a regression layers (Figure 8.1 (g)) to train for action classification and micro-tube regression. The outputs of the network are $C \times M$ softmax classification scores and $8 \times M$ bounding box coordinates for M predicted micro-tubes (Figure 8.1 (h)) and C action categories. In all our experiments, at train time we use $M = 2000$ and test time $M = 300$.

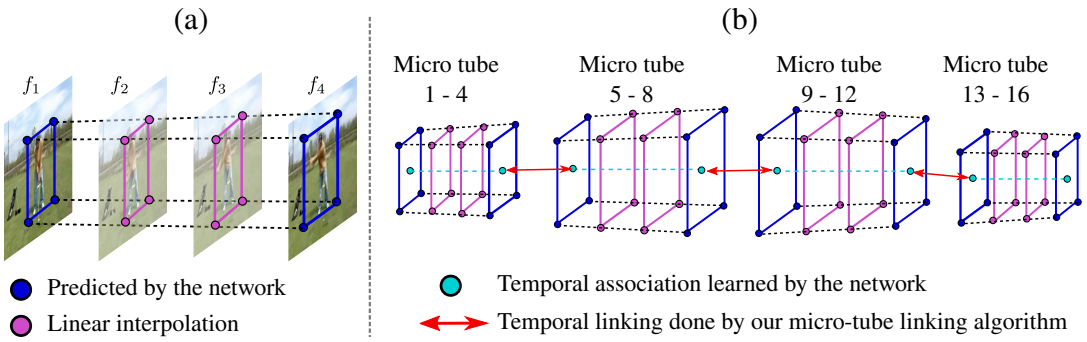


Figure 8.2: ***(a)*** *Action micro-tube generation by linear interpolation of coordinates of predicted bounding boxes. The blue circles denote the 8 (x, y) coordinate values of bounding boxes predicted by our network for a pair of successive test video frames. The first frame in the pair belongs to time $t = 1$ and the seconds frame belongs to time $t = 4$. We generate the coordinates of detection bounding boxes (pink circles) for intermediate frames (i.e. frame 2 and 3) by linear interpolation of the predicted coordinates. Thus, at test time, our model processes a pair of successive frames by skipping two intermediate frames (in this example frame 2 and 3) which substantially reduces the computing cost and time and speeds up the detection process.* ***(b)*** *Action tube generation by linking micro-tubes generated by* ***(a)****. Note that, for a video sequence with T frames, our model needs to process only $T/2$ frames and the micro-tube linking algorithm requires to connect only $(T/2) - 2$ frames.*

8.3 Bounding box interpolation and action tube generation

At test time, for a input pair of successive video frames (f_t and $f_{t+\Delta}$), the network outputs action classification scores and coordinate values for 300 micro-tubes (Section 8.2.3). For example, in Figure 8.2 (a), we can visualise that for an input pair $(F1, F4)$ our network predicts an action micro-tube (depicted by the 2 blue bounding boxes) with 8 coordinate values (blue circles). In practice, the network predicts such 300 micro-tubes for an input pair. Note that, yet we do not have detection bounding boxes for intermediate frames, in this example those are frames $F2$ and $F3$. We generate the detection boxes for the intermediate frames using a simple but elegant box interpolation algorithm which is explained in the next section.

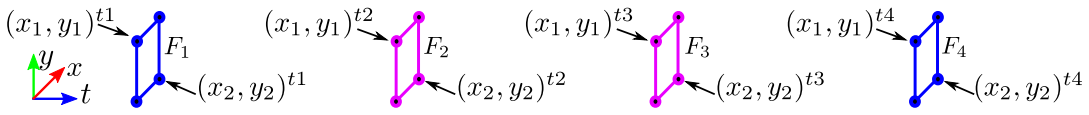


Figure 8.3: Generation of detection bounding boxes for intermediate frames using linear interpolation.

8.3.1 Bounding box interpolation

As depicted in Figure 8.3, the 8 predicted bounding box coordinates $[(x_1, y_1, x_2, y_2)^{t_1}]$ and $[(x_1, y_1, x_2, y_2)^{t_4}]$ (shown in blue) which are known, and we would like to infer the unknowns $[(x_1, y_1, x_2, y_2)^{t_2}]$ and $[(x_1, y_1, x_2, y_2)^{t_3}]$ (shown in pink). Using simple linear interpolation we can compute the unknowns using the following equation:

$$x_1^{t2} = x_1^{t1}(1 - \frac{t_2 - t_1}{t_4 - t_1}) + x_1^{t4}(1 - \frac{t_2 - t_1}{t_4 - t_1}) \tag{8.1}$$

$$y_1^{t2} = y_1^{t1}(1 - \frac{t_2 - t_1}{t_4 - t_1}) + y_1^{t4}(1 - \frac{t_2 - t_1}{t_4 - t_1}) \tag{8.2}$$

$$x_1^{t3} = x_1^{t1}(1 - \frac{t_3 - t_1}{t_4 - t_1}) + x_1^{t4}(1 - \frac{t_3 - t_1}{t_4 - t_1}) \tag{8.3}$$

$$y_1^{t3} = y_1^{t1}(1 - \frac{t_3 - t_1}{t_4 - t_1}) + y_1^{t4}(1 - \frac{t_3 - t_1}{t_4 - t_1}) \tag{8.4}$$

Similarly we can compute $(x_2, y_2)^{t2}$ and $(x_2, y_2)^{t3}$.

8.3.2 Action tube generation

Once the bounding-boxes for intermediate frames are generated usning the linear interpolation algorithm (Section 8.3.1) for each micro-tubes, we can then temporally link them (Figure 8.2 (b)) using action tube generation algorithm (Section 7.5) presented in the previous chapter.

The micro-tube linking algorithm presented in Section 7.5 requires lesser iterations than the one in Section 5.3.4, i.e., it requires $(T/2 - 1)$ as compared to $(T - 1)$ iterations (where T is the number of frames in a video). Unlike in Section 7.5 where micro-tubes are extracted for consecutive frame pairs (i.e. $\Delta = 1$), in this chapter, we use long distance frame pairs (i.e. $\Delta = 3$) at test time which allow our micro-tube linking algorithm to iterate only $(T/4 - 1)$ times to link all the micro-tubes associated with an action instance within a video. Thus, this new tube generation approach is relatively faster and requires less compute time and resources than the one in Section 7.5. For instance, as depicted in Figure 8.2 (b), tube generation algorithm presented here requires only 3 iterations to link the micro-tubes of a video having 16 frames. Whereas, the algorithm in Section 5.3.4 needs 15 iterations and the one in Section 7.5 requires 7 iterations to link them in time.

We extend the action tube generation (i.e. micro-tube linking) module presented in Section 7.5. In the 1^{st} pass of DP (dynamic programming), class-

specific action paths $\mathbf{p}_c = \{m_t, \quad t \in I = \{4, 8, ..., T-4\}\}$, spanning the entire video duration are obtained by maximising via dynamic programming:

$$E(\mathbf{p}_c) = \sum_{t \in I} s_c(m_t) + \lambda_o \sum_{t \in I} \psi_o \left({b^2}_{m_t}, {b^1}_{m_{t+1}} \right) \tag{8.5}$$

where m denotes a micro-tube comprised of two bounding boxes b^1 and b^2, $s_c(m_t)$ represents the softmax score (Section 8.2.3) of the predicted micro-tube m at time step t, $\psi_o \left({b^2}_{m_t}, {b^1}_{m_{t+1}} \right)$ is the IoU between the 2^{nd} detection box ${b^2}_{m_t}$ of micro-tube m_t and the first detection box $b^1_{m_{t+1}}$ of micro-tube m_{t+1}. λ_o is a scalar parameter weighting the relative importance of the pairwise term. The above energy maximisation (Equation 8.5) gives us action tubes which are temporally untrimmed. For temporal localisation of actions, we trim the action tubes using a similar approach as in Section 7.5 using a 2^{dn} pass of DP (or a temporal label smoothing approach).

8.4 Train and test data sampling schemes

To train and test our proposed model, we need to pass pairs of successive RGB video frames (f_t, $f_{t+\Delta}$) and a set of 5 stacked optical flow maps as inputs to the network (Section 8.6.4). At training time, we generate long- and short-distance training pairs using different Δ values.

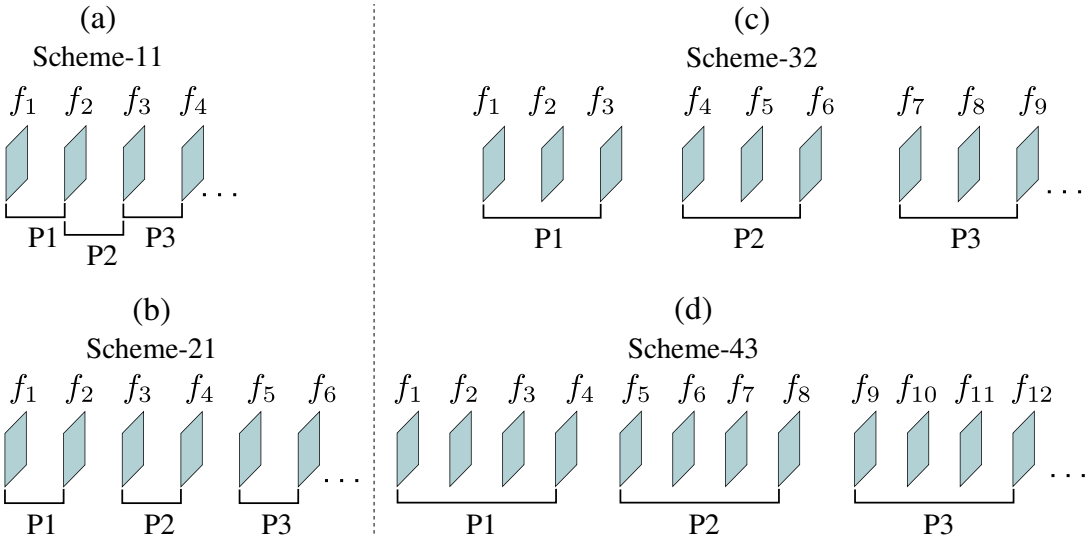


Figure 8.4: Train and test time data sampling schemes.

Figure 8.4 shows four different data sampling schemes, in which F_i denotes a RGB video frame where i is the frame index and P_j denotes a training pair

where j is the pair index. *Scheme-11* and *scheme-21* (Figure 8.4(a) & (b)) generate short-distance pairs using Δ value 1. Whereas, *Scheme-32* and *scheme-43* (Figure 8.4(c) & (d)) generate relatively long-distance pairs using Δ values 2 and 3 respectively.

8.4.1 Training data of RGB video frames

To training our model on *J-HMDB-21* [18] dataset, we generate train sets using data sampling schemes: *scheme-11*, *-32* and *-43* which give rise to a total 68796, 69350 and 69054 training pairs for split 1, 2 and 3 respectively. For *UCF-101-24* dataset (split-1), we create a training set by generating pairs using *scheme-21*, *-43* which jointly give $509,940$ training pairs.

8.4.2 Training data of optical-flow maps

We capture the local motion patterns (of human actions) between two consecutive video frames by computing optical flow maps (see Section 2.7 and A.1). For training pairs generated using *scheme-11* and *-21* ($\Delta = 1$) (Section 8.4.1), we select the corresponding five flow-maps following a sequence:
$\{f_{t-1}, f_t, f_{t+\Delta}, f_{t+\Delta+1}, f_{t+\Delta+2}\}$,
for *scheme-32* ($\Delta = 2$), we follow a sequence:
$\{f_{t-1}, f_t, f_{t+1}, f_{t+\Delta}, f_{t+\Delta+1}\}$,
and for *scheme-43* ($\Delta = 3$):
$\{f_{t-1},\ f_t,\ f_{t+1},\ f_{t+2},\ f_{t+\Delta}\}$.

8.4.3 Test time data sampling

At test time, we sample long-distance RGB frame pairs using *scheme-43* (Section 8.4.1) and follow the corresponding sequence (Section 8.4.2) to select five optical-flow maps.

8.5 Model Evaluation

Experimental settings. Please refer Section 4.2.3 for the frame-mAP and video-mAP evaluation metrics used in this section. We follow the training data sampling strategy as in [38], in addition, for training on long distance pairs we introduce a new scheme *scheme-43*. Throughout this section we use the following naming conventions to denote models trained on training sets generated using

Table 8.1: Impact of adding motion stream on action detection performance, J-HMDB-21 (*) dataset.

	Video-mAP					Frame-mAP
δ	0.1	0.2	0.3	0.4	0.5	0.5
AMTnet	57.79	57.76	57.68	56.79	55.31	45.0
AMTnet-Flow	**64.64**	**64.64**	**64.57**	**64.01**	**62.68**	**52.08**

(*)Trained *model-11+32*, action tubes generated using micro-tube linking algorithm as in [38].

Table 8.2: Impact of adding motion stream on action detection performance, UCF-101-24 (*) dataset.

	Video-mAP					Frame-mAP
δ	0.1	0.2	0.3	0.4	0.5	0.5
AMTnet	71.3	63.06	51.57	43.10	33.06	54.91
AMTnet-Flow	**79.65**	**71.51**	**59.79**	**47.28**	**35.77**	**62.70**

(*)Trained *model-21*, action tubes generated using 2PDP with cross validated alphas as in [33].

different sampling strategies: a) a model trained on training data generated using *scheme-11* and *-32* [38] is referred as *model-11+32*; b) *model-11+32+43* refers the model trained using *scheme-11*, *-32* and *-43*; similarly, c) *model-21* and d) *model-21+43*. For more details on the train and test data sampling schemes please refer Section 8.4.

8.5.1 Impact of adding motion stream on action detection performance

We observe a significant improvement in action detection performance by adding an extra flow stream to the AMTnet architecture. For this experiment, we first reproduce the results of AMTnet [38] on both J-HMDB-21 and UCF-101-24 datasets following their train and test time settings. Then, we introduce our motion-stream alongside the two spatial streams of AMTnet, and train the model on both the datasets following AMTnet's train and test time settings. In Table 8.1 and 8.2, we refer these models as AMTnet and AMTnet-Flow where the video- and frame-mAPs at different IoU thresholds (δ) for these two models are shown. Note that for J-HMDB, the video- and frame-mAP are improved by **7.37%** and **7.08%** at $\delta = 0.5$ (Table 8.1). For UCF-101-24, video-mAP (at $\delta = 0.2$) and frame-mAP (at $\delta = 0.5$) are improved by**8.45%** and **7.79%** (Table 8.2).

Table 8.3: Impact of bounding box interpolation on action detection performance (video-mAP).

	J-HMDB			UCF-101		
δ	0.2	0.4	0.5	0.2	0.5	0.5:0.95
w/o interpolation	**64.73**	**64.32**	**62.67**	**71.57**	35.77	10.9
with interpolation	64.63	64.27	61.97	69.86	**37.15**	**11.9**

Table 8.4: Impact of bounding box interpolation on action detection speed (seconds / video).

	J-HMDB	UCF-101
w/o interpolation	11.9	61.25
with interpolation	**5.95**	**30.62**

8.5.2 Impact of bounding box interpolation on detection performance and speed

For this experiment, we train our model on J-HMDB-21 and UCF-101-24 datasets on both short and long distance pairs. In Section 8.4, we discuss in detail the different training data sampling schemes used to generate the training data for each model in this experiment. At test time, we generate two sets of detection results to demonstrate the efficacy of our bounding box interpolation algorithm (Section 8.3). First set of micro-tubes are extracted by passing test pairs f_t and $f_{t+\Delta}$ where $\Delta = 1$ (short distance pairs). Second set of micro-tubes are extracted by passing test pairs where $\Delta = 3$ (long distance pairs). For micro-tubes extracted using long-distance pairs, we fill in the intermediate frames' bounding boxes using our box interpolation algorithm (Section 8.3). We denote these two sets of results in Table 8.3 and Table 8.4 as "w/o interpolation" and "with interpolation". Our bounding box interpolation algorithm improves the test time detection speed significantly without affecting the detection performance (Table 8.4). For J-HMDB-21, it improves the detection speed from 11.9 sec/vid (seconds per video) to **5.95** sec/vid, and for UCF-101, from 61.25 sec/vid to **30.62** sec/vid. Here detection speed denotes the compute time required for micro-tube extraction. Interestingly, for UCF-101, it improves the video-mAP at higher thresholds, i.e. we observe a gain in the mAP by **1.38%** and **1.0%** at $\delta = 0.5$ and $0.5 : 0.95$ respectively (Table 8.3). Note that, the test video clips in UCF-101-24 have relatively longer duration (on average 175 frames) than J-HMDB-21 (on average 25 frames). The relatively shorter test video clips might be the reason that our bounding box interpolation algorithm could not exhibit performance boost on J-HMDB-21. We believe that, it will work gracefully for

Table 8.5: Impact of our action tube generation algorithm on compute time (in seconds per video).

–	Saha *et al.* [33]	Ours	–	Saha *et al.* [33]	Ours
J-HMDB-21	0.38	**0.16**	UCF-101-24	0.96	**0.47**

Table 8.6: Comparison with the state-of-art on J-HMDB-21 dataset.

	Video-mAP		Frame-mAP
δ	0.2	0.5	0.5
Gkioxari *et al.* [14]	–	53.3	36.2
Wang *et al.* [134]	–	56.4	39.9
Weinzaepfel *et al.* [20]	–	60.7	45.8
Yu *et al.* [26]	–	–	–
Saha *et al.* [33]	72.63	71.50	–
Saha *et al.* [38]	57.76	55.31	45.0
Peng *et al.* [32]	**74.3**	**73.1**	**58.5**
Ours (w/o interpolation)	64.73	62.67	52.08
Ours (with interpolation)	64.63	61.97	52.73

even longer sequences of videos where spatial locations of actions do not vary rapidly within shorter time span (e.g. within 5 or 10 frames).

8.5.3 Impact of our action tube generation algorithm on compute time

To evaluate our action tube generation method (Section 8.3), we compare the detection performance and compute time requirement of our algorithm with [33]. In Table 8.3, the "w/o interpolation" refers to the results obtained using the action-tube generation algorithm [33], and "with interpolation" denotes the results generated by our tube building approach. Remarkably, with relatively lesser number of iterations ($T/4-1$) as compared to ($T-1$) iterations required by [33], our action tube generation algorithm improves the video-mAP at higher thresholds on UCF-101. Moreover, our method exhibits excellent mean detection speed (Table 8.5) of 0.16 sec/vid and 0.47 sec/vid for J-HMDB and UCF-101 respectively. In particular, it improves the speed by **0.22** (for J-HMDB) and **0.49** (for UCF-101) sec/vid over [33]'s approach.

8.5.4 Comparison with the state-of-art

In this section, we compare the action detection performance of the proposed approach with the state-of-the-art methods (see Table 8.6 and Table 8.7). The proposed approach outperforms the state-of-the-art methods [14, 20, 26, 38, 134]

Table 8.7: Comparison with the state-of-art on UCF-101-24 dataset.

	Video-mAP				Frame-mAP
δ	0.1	0.2	0.5	0.5:0.95	0.5
Gkioxari *et al.* [14]	–	–	–	–	–
Wang *et al.* [134]	–	–	–	–	–
Weinzaepfel *et al.* [20]	51.7	46.8	–	–	35.84
Yu *et al.* [26]	42.8	26.5	–	–	–
Saha *et al.* [33]	76.12	66.36	34.82	–	–
Saha *et al.* [38]	71.3	63.06	33.06	10.72	54.91
Peng *et al.* [32]	77.31	**72.86**	30.87	07.11	**65.73**
Ours (w/o interpolation)	**79.65**	71.57	35.77	10.96	62.7
Ours (with interpolation)	78.24	69.86	**37.15**	**11.9**	62.86

in both frame- and video-mAP on two benchmark datasets. In addition, our approach outperforms the two close competitors [32, 33] in spatio-temporal detection (video-mAP) on UCF-101. More specifically, it outperforms [33] (in video-mAP for UCF-101) by 5.21% and 2.33% at IoU threshold (δ) values 0.2 and 0.5 respectively. Although, on lower δ values (0.2 and 0.3) our approach shows comparable results to [32], on higher δs (0.5 and 0.5 : 0.95), it achieves superior performance with a gain in the video-mAP by **6.28%** and **4.79%** respectively. A lower detection performance of our approach on J-HMDB-21 as compared to [32,33] is due to the fact that J-HMDB is relatively smaller dataset and our network has large number of model parameters than [32,33] which might cause overfitting. In Section 8.5.5, we present the class-specific frame- and video-mAP comparison with the state-of-art.

8.5.5 Comparison of class-specific frame- and video-level APs

In this section, we compare the class-specific frame- and video-level AP (average precision) with the state-of-the-art [14, 20, 32, 33, 38, 134]. We report frame- and video-AP on J-HMDB-21 and video-AP on UCF-101-24 datasets. In Table 8.8, 8.9 and 8.10, we denote the highest APs with red and second highest APs with blue colour.

J-HMDB-21. For J-HMDB-21, both frame and video-APs are computed at IoU threshold $\delta = 0.5$ and results are averaged over the 3 splits of J-HMDB-21. Table 8.8 presents the per class frame-AP comparison with the state-of-the-art [14,20,32,38,134]. Our method achieves the highest frame-AP for *five action categories* with an improvement of **8.4%**, **7.6%**, **6.1%**, **3.6%** and **0.1%** for

Table 8.8: J-HMDB-21: per class frame-AP comparison with the state-of-the-art.

Frame-AP(%) at $\delta = 0.5$

	brushHair	catch	clap	climbStairs	golf	jump
Gkioxari *et al.* [14]	65.2	18.3	38.1	39.0	79.4	7.3
Wang *et al.* [134]	60.1	34.2	56.4	38.9	83.1	10.8
Weinzaepfel *et al.* [20]	73.3	34.0	40.8	56.8	93.9	5.9
Saha *et al.* [38]	43.7	43.6	33.0	61.5	91.8	5.6
Peng *et al.* [32]	75.8	38.4	62.2	62.4	99.6	12.7
Ours	50.9	49.7	34.5	70.8	93.7	12.2

	run	shootBall	shootBow	shootGun	sit	stand
Gkioxari *et al.* [14]	11.6	5.6	66.8	27.0	32.1	34.2
Wang *et al.* [134]	19.8	11.6	78.0	50.6	10.9	43.0
Weinzaepfel *et al.* [20]	21.1	23.9	85.6	37.8	34.9	49.2
Saha *et al.* [38]	32.3	33.3	81.4	55.1	12.4	14.7
Peng *et al.* [32]	38.1	52.8	90.8	62.7	33.6	48.9
Ours	45.7	42.6	88.3	62.8	23.3	45.0

	kickBall	pick	pour	pullup	push
Gkioxari *et al.* [14]	9.4	25.2	80.2	82.8	33.6
Wang *et al.* [134]	24.5	38.5	71.5	67.5	21.3
Weinzaepfel *et al.* [20]	13.8	38.5	88.1	89.4	60.5
Saha *et al.* [38]	23.8	31.5	91.8	84.1	73.1
Peng *et al.* [32]	35.1	57.8	96.8	97.3	79.6
Ours	38.7	41.9	92.1	87.4	75.1

	swingBaseball	throw	walk	wave	**mAP**
Gkioxari *et al.* [14]	33.6	15.5	34.0	21.9	36.2
Wang *et al.* [134]	48.9	26.5	25.2	15.8	39.9
Weinzaepfel *et al.* [20]	36.7	16.8	40.5	20.5	45.8
Saha *et al.* [38]	56.3	22.2	24.7	29.4	45.0
Peng *et al.* [32]	62.2	25.6	59.7	37.1	58.5
Ours	48.4	23.9	42.5	24.0	52.7

action class *climbStairs*, *run*, *catch*, *kickBall* and *shootGun* respectively. For another 7 action categories, it achieves the second highest AP with an improvement of **9.3%**, **3.4%**, **2.7%**, **2%**, **2%**, **1.4%** and **0.3%** for class *shootBall*, *pick*, *shootBow*, *walk*, *push*, *jump* and *pour*s respectively. Table 8.9 presents the per class video-AP comparison with [14, 38, 134]. We could not consider methods [20, 32, 33] in our class-specific video-AP comparison, as they do not report per class video-AP for J-HMDB-21 in their respective papers. Among 21 action classes of J-HMDB-21, our method shows superior video-level performance for 7 classes with an improvement of **16.5%**, **13.4%**, **9%**, **7.4%**, **3.7%**, **2.5%** and **1.8%** for action class *shootBall*, *climbStairs*, *run*, *catch*, *push*, *shootBow* and *kickBall* respectively.

Table 8.9: J-HMDB-21: per class video-AP comparison with the state-of-the-art.

Video-AP(%)(*) at $\delta = 0.5$

	brushHair	catch	clap	climbStairs	golf	jump
Gkioxari *et al.* [14]	79.1	33.4	53.9	60.3	99.3	18.4
Wang *et al.* [134]	76.4	49.7	80.3	43.0	92.5	24.2
Saha *et al.* [38]	51.9	54.5	41.2	66.6	94.8	7.8
Ours	65.4	61.9	46.4	80.0	92.5	20.4

	run	shootBall	shootBow	shootGun	sit	stand
Gkioxari *et al.* [14]	24.6	13.7	92.9	42.3	67.2	57.6
Wang *et al.* [134]	35.7	27.0	88.8	76.9	29.8	68.6
Saha *et al.* [38]	49.0	37.4	92.7	75.8	21.6	27.1
Ours	58.0	53.9	95.4	73.9	40.7	49.7

	kickBall	pick	pour	pullup	push	
Gkioxari *et al.* [14]	26.2	42.0	92.8	98.1	29.6	
Wang *et al.* [134]	57.7	70.5	78.7	77.2	31.7	
Saha *et al.* [38]	48.7	33.7	97.6	92.5	87.6	
Ours	59.5	47.9	97.4	92.6	91.3	

	swingBaseball	throw	walk	wave	**mAP**
Gkioxari *et al.* [14]	66.5	27.9	58.9	35.8	53.3
Wang *et al.* [134]	72.8	31.5	44.4	26.2	56.4
Saha *et al.* [38]	73.3	24.3	37.7	44.7	55.31
Ours	62.6	27.8	54.8	43.0	62.67

UCF-101-24. UCF-101-24 is a temporally untrimmed action detection dataset for which we have both spatial (frame-level bounding boxes) and temporal (start and end of each action instance) ground truths available. Thus, it is best suited for evaluating the spatio-temporal action detection performance. A frame-AP metric measures the quality of a detection algorithm only against spatial action detection task. Therefore, we use video-AP to evaluate the spatio-temporal action detection performance. We compare the per-class video-AP of our method with the state-of-the-art [33,38]. Again, we could not consider methods [20,32] in our class-specific video-AP comparison, as they do not report per class video-AP for UCF-101-24 in their respective papers. For UCF-101-24, we report video-AP at standard IoU threshold $\delta = 0.2$.

Table 8.10 shows the per class video-AP comparison with the state-of-the-art. Among 24 action classes, our method achieves the highest video-AP for 13 classes with an overall gain in the video-mAP of **5.21%**. Note that, our method improves the quality of the spatio-temporal action detection with a large margin. It can be observed by looking at the video-APs of those classes which contain significantly temporally untrimmed test video clips. For example, action classes *volleyball spiking*, *cricket bowling* and *basketball dunk* have videos (in UCF-101

Table 8.10: UCF-101-24: per class video-AP comparison with the state-of-the-art.

	BaBa	BaDu	Bi	ClDi	CrBo	Di	Fe	FlGy
Saha *et al.* [33]	36.7	48.3	60.4	73.2	19.9	96.6	88.0	99.7
Saha *et al.* [38]	29.8	40.9	63.8	44.9	15.2	91.2	89.3	94.2
Ours	26.8	75.0	60.4	91.4	31.8	97.4	90.4	99.2

	GoSw	HoRi	IcDa	LoJu	PoVa	RoCl	SaSp	SkBo
Saha *et al.* [33]	66.5	94.1	62.5	55.7	72.6	89.6	57.5	85.0
Saha *et al.* [38]	70.7	92.3	70.5	50.7	71.0	97.3	31.4	72.0
Ours	80.3	92.8	74.0	80.4	87.0	97.1	37.6	76.6

	Sk	SkJe	SoJu	Su	TeSw	TrJu	VoSp	WaDo	**mAP**
Saha *et al.* [33]	78.9	92.8	86.4	61.3	32.6	51.3	15.9	75.2	66.36
Saha *et al.* [38]	74.0	95.9	80.2	67.6	40.4	42.0	13.0	75.0	63.06
Ours	76.7	96.2	93.2	62.8	32.0	47.4	31.8	78.0	71.57

BaBa : Basketball, BaDu : BasketballDunk, Bi : Biking, ClDi : CliffDiving,
CrBo : CricketBowling, Di : Diving, Fe : Fencing, FlGy : FloorGymnastics,
GoSw : GolfSwing, HoRi : HorseRiding, IcDa : IceDancing, LoJu : LongJump,
PoVa : PoleVault, RoCl : RopeClimbing, SaSp : SalsaSpin, SkBo : SkateBoarding,
Sk : Skiing, SkJe : Skijet, SoJu : SoccerJuggling, Su : Surfing, TeSw : TennisSwing,
TrJu : TrampolineJumping, VoSp : VolleyballSpiking, WaDo : WalkingWithDog.

testsplit-1) which contain action instances less than **50%** of the entire video on average. In other words, a test video belongs to these classes do not have any action instance present more than 50% of the entire video on average. Our method improves the video-AP for these classes by a large margin of **26.7%**, **15.9%** and **11.9%** for classes *basketball dunk*, *volleyball spiking* and *cricket bowling* respectively. Similarly, we have another two classes *cliff diving* and *diving* which have highly temporally untrimmed video clips in the testset. For these two classes, we report an improvement of **18.2%** and **0.8%** in video-AP.

Discussion. For some instances, Peng *et al.*'s [32] approach exhibits better detection performance on UCF-101-24 than the proposed solution (Table 8.7). However, we argue that their approach is expensive. Firstly, they train two separate networks for appearance and motion streams independently, they name their network as *"two-stream R-CNNs"*. Secondly, they finetune these two networks on *multi-region* based action proposals using the learned weights of the two-stream R-CNNs (for more detail, please refer to [32]). Note that, their framework follows a *multi-stage* training strategy. Moreover, they follow *multi-scale* training and testing schemes in which at train (or test) time, for each original video frame

three re-scaled versions of that frame are generated, and subsequently they are used for training or testing. It is quite apparent that Peng *et al.*'s approach is computationally expensive both at training and testing.

In contrast, we train both the appearance and motion streams jointly and the proposed approach follows a *single-stage* training i.e. our network requires only one time training and does not require any finetuning as in [32]. At test time, the network receives a pair of successive video frames (f_t and $f_{t+\Delta}$ where $\Delta = 4$, see Figure 8.2 (a)) as input, and predicts pairs of bounding boxes (called action micro-tubes [38], see Figure 8.2 (b)) which are linked in time. Thus, our model requires to process only 50% of the entire video frames which significantly reduces the computing time and cost. Moreover, we follow single-scale training and testing schemes i.e., we use each video frame only once as compared to [32] which uses one frame thrice (at 3 different scales). With these aforementioned properties i.e., an end-to-end trainable network and relatively low computing requirements, the proposed approach outperforms [32] at higher IoU thresholds (δ) (Table 8.7) on one of the largest action detection datasets (UCF-101-24 [17]) to date.

One of the standard techniques for reducing overfitting is data augmentation. As discussed above, Peng *et al.* [32] augment their training data with three re-scaled versions of the original video frame (i.e. the multi-scale training), and use three extra sets of region proposals alongside the original set of RPN proposals (i.e. the multi-region based action proposals). To train their network on these additional proposals, they add three extra ROI pooling layers (alongside the original RPN-based ROI pooling layer). Although, their multi-scale and multi-region based training approach helps in improving the detection performance on J-HMDB-21, they are expensive. As J-HMDB-21 is relatively smaller dataset as compared to UCF-101-24 (Chapter 4), overfitting might happen during training our network (on J-HMDB-21) due to the fact that we do not use any expensive data augmentation such as multi-scale technique used in [32], resulting relatively lower detection performance. It is interesting to see how our AMTnet-Flow behaves with multi-scale and multi-region based training. We keep this extension as a future work.

8.6 Implementation details

8.6.1 Hardware and software platform

For all our experiments, we use an Ubuntu 14.0 LTS based Dell workstation equipped with an Intel Xeon CPU @ 3.20GHz, 64 GB of RAM and a Nvidia GeForce GTX TITAN X GPU. To implement our proposed model, we use the scientific computing framework Torch 7.0 [164] and scripting language LuaJIT [167] . We implement our action-tube generation algorithm in MATLAB. During implementation, we refer the Torch codebase developed by [38]. For bilinear interpolation, we use the publicly available Torch repository [166].

8.6.2 Data preprocessing and train data augmentation

Both at train and test time, we preprocess the training data by first scaling each video frame to 800×600 pixel resolution [29], and then subtracting the VGG image mean $\{103.939, 116.779, 123.68\}$ from each RGB frame, and optical-flow mean $\{128, 128, 128\}$ from each flow-map [32]. We use horizontal flipping of each video frame with a probability 0.5 to augment our training sets.

8.6.3 Network weight initialisation and optimisation

We initialise the weights of the convolutional layers with VGG-16 pretrained ImageNet [153] weights. For the rest of the network layers, weights are initialised by drawing random samples from a Gaussian distribution with a standard deviation 0.01. We train our network end-to-end as in [38]. At each training iteration, we update the weights of the convolutional layers of both appearance- and motion-streams using stochastic gradient descent (SGD) with a momentum 0.9; the Adam optimisation algorithm [162] is used to update the weights of the rest of the layers. For Adam, we use parameter values $\beta_1 = 0.9$, $\beta_2 = 0.99$ and a learning rate of 1×10^{-6}. In the first training epoch, we update the weights of all the network layers excluding the three parallel CNNs. During the first epoch we freeze the weights of the convolutional layers. From second epoch onward, we update the weights of all layers including the appearance- and motion-stream CNNs. For computational efficiency, the first four layers of the three CNNs are not fine-tuned.

8.6.4 Architectural modification for motion stream

We modify the architecture of the 1^{st} convolutional layer of VGG-16 to pass five stacked optical-flow maps (Section A.1) as inputs to the motion stream. The original VGG-16 can receive input a tensor of shape $[3 \times H \times W]$, i.e., a single video frame (H and W are the height and width of the input frame). We modify the architecture of the 1^{st} convolutional layer of VGG-16 such that, it can receive an input of shape $[(3 \times 5) \times H \times W]$, i.e., a set of 5 stacked optical-flow maps. There are 64 convolutional kernels (filters) of dimension $3 \times 3 \times 3$ in the 1^{st} convolutional layer of VGG-16, with the above modification, now we have 64 convolutional kernels of dimension $15 \times 3 \times 3$. We initialise the weights of these new 64 convolutional kernels with the VGG-16 pretrained ImageNet [153] weights for the 1st layer by simply duplicating the weights for 5 times.

8.6.5 Training iterations

We train our model for $320k$ iterations on J-HMDB-21 and $1170k$ iterations on UCF-101-24. The training durations are 3 and 7 GPU days for J-HMDB and UCF-101 respectively. We follow a same train and test time 3D-ROI sampling strategies as used in Section 7.3.3.

Chapter 9

Conclusions and Future Research Directions

In this thesis, we have introduced a variety of novel deep learning based frameworks (Chapter 5, 7 & 8) for action detection in videos. In addition, we have addressed the problem of space-time action instance segmentation (Chapter 6). An instance segmentation method can provide more accurate localisation results (than action detection) which are highly beneficial for real-world applications such as self-driving cars. This chapter is organised as follows. Firstly, we summarise the contributions of the thesis in Section 9.1. Secondly, we conclude by bringing forward some of the prominent research directions for future work (Section 9.2).

9.1 Summary of contributions of the thesis

9.1.1 Action detection using frame-level deep features

In Chapter 5, we have proposed a novel human action detection approach by leveraging frame-level deep features (Section 2.1) and a two-pass Viterbi (dynamic programming) algorithm (Section A.5). The proposed method addresses in a coherent framework the challenges involved in concurrent multiple human action recognition, spatial localisation and temporal detection, thanks to a novel deep learning strategy for simultaneous detection and classification of region proposals and an improved action tube generation approach. Our method significantly outperforms the previous state-of-the-art on the most challenging benchmark datasets, for its capability of handling multiple concurrent action instances and temporally untrimmed videos. Our experimental results have demonstrated

quantitatively that: (a) a two-stream based deep architecture coupled with our novel test-time fusion technique (for combining appearance and motion cues) significantly improves detection performance; (b) our label smoothing approach (2^{nd} pass of dynamic programming) works gracefully on highly temporally untrimmed videos and improves the detection results; (c) deep feature based supervised region proposals show better recall-to-IoU [29] than unsupervised proposals, (d) using a single-stage detection framework as compared to a multi-stage pipeline helps reducing the computational time and cost by a large margin, and (e) deep features exhibit better representational power than shallow features in action recognition task.

9.1.2 Action instance segmentation

Emerging real-world applications (such as self-driving cars) require autonomous systems which can localise action instances at finer pixel-level as compared to coarse bounding-box level detections. The existing action detection approaches [14, 20, 32] (including those presented in Chapter 5, 7 & 8) are limited to localising actions at bounding-box level and they work in a setting where the videos are temporally trimmed as per the temporal extents of actions and they contain action instances belong to a single action category.

In Chapter 6, we have addressed these aforementioned drawbacks in the existing action detectors by introducing a novel spatio-temporal action instance segmentation approach which can: (a) delineate action instances in both space and time at pixel-level and classify each instance by assigning (to it) a class- and instance-aware label, and (b) perform spatio-temporal action localisation on temporally untrimmed videos containing multiple co-occurring actions belonging to any action categories. We have validated the proposed method on the challenging LIRIS-HARL D2 [128] activity detection dataset which contains multiple concurrent actions, with instances belong to same and/or different action class happening at the same time, and where all videos are temporally untrimmed. Our proposed pipeline achieved new benchmark performance which is 14.3 times better than the previous top performer. To the best of our knowledge, we are the first to qualitatively demonstrate the action instance segmentation results on the LIRIS-HARL and UCF-101-24 datasets.

9.1.3 Action detection using video-level deep features

The most recent deep learning based action detection approaches [14, 20, 32] (including those presented in Chapter 5 & 6) follow a frame-level action representation and training. In such methods, rather than learning to classify and regress 3D region proposals, the network learns to classify and regress 2D proposals. Such frame-level solutions are suboptimal and heavily rely on a post-processing step to temporally link frame-level detections to build action tubes. In Chapter 7, we have departed from such current practice in action detection (i.e. frame-level action representation and training) to take a step towards deep network architectures able to classify and regress whole video subsets (i.e. video-level representation and training). In particular, we have proposed a novel deep net framework able to regress and classify 3D region proposals spanning two successive video frames, effectively encoding the temporal aspect of actions using just raw RGB values. We have termed this new deep network as *"AMTnet"* which is end-to-end trainable and can be jointly optimised for region proposal generation and action detection objectives using a single step of optimisation. At test time the network predicts 'micro-tubes' spanning two frames, which are linked up into complete action tubes via a new algorithm of our design. Promising results confirm that AMTnet does indeed outperform the state-of-the-art when relying purely on appearance.

In Chapter 8, we have extended AMTnet deep architecture to improve the action representation by integrating the complementary optical flow features in the same framework. We have termed the extended AMTnet as *"AMTnet-Flow"*. Unlike AMTnet training, we have trained AMTnet-Flow on both nearby and widely separated frame pairs (i.e. frame pairs with long and short inter-frame distance) and at test-time, detection micro-tubes have been extracted on long distance pairs. Our experiments quantitatively demonstrate that: (a) the action detection performance is significantly improved by integrating optical flow based deep features in the same AMTnet framework; (b) the micro-tube extraction at test-time is made relatively faster with our new bounding-box interpolation algorithm which populates detection bounding-boxes for the intermediate frames by linear interpolation of box coordinates; (c) our new micro-tube linking algorithm to link relatively longer micro-tubes further reduces the computing time required for tube generation. We have further demonstrated significant improvements in frame-level and video-level mAP over the existing state-of-the-art approaches [14,20,26,38,134]. Most noticeably, AMTnet-Flow outperforms the top competitor [32] in video-mAP by a large margin of **6.28%** and **4.79%** on higher

IoU thresholds 0.5 and 0.5 : 0.95 respectively on UCF-101-24 action detection benchmark.

9.2 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.

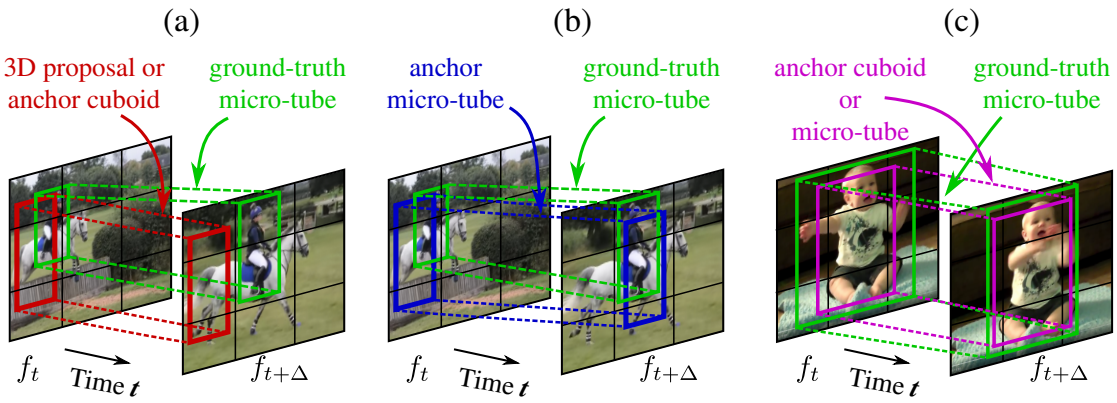


*Figure 9.1: Illustrating the key limitation of 3D action proposals or anchor cuboids using a "dynamic" action like "horse riding". **(a)** A horse rider changes its location from frame f_t to $f_{t+\Delta}$ as shown by the ground truth bounding boxes (in green). As the anchor cuboid generation is constrained 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, we envision an anchor micro-tube proposal generator would be much more flexible, as it would efficiently explore the video search space. As a result, the anchor micro-tube proposal (in blue) would exhibit higher overlap with the ground truth. **(c)** For "static" actions like "clap" in which the actor does not change location over time, an anchor cuboid and an anchor micro-tube would have the same spatiotemporal bounds.*

9.2.1 Improving 3D action proposal quality

In Chapter 7 & 8, we have generated action proposals by extending 2D action proposals (anchor boxes for images) to 3D proposals (anchor cuboids for videos) (see Figure 9.1 **(a)**). The main limitation of this approach is it cannot provide an optimal set of training hypotheses, as 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. The video search space grows exponentially as the number of video frames f increases. Furthermore, anchor cuboids are very limiting for action detection purposes. Whereas they can be suitable for "static" actions (e.g. "handshake" or

"clap", in which the spatial location of the actor(s) does not vary over time), they are most inappropriate for "dynamic" ones (e.g. "horse riding", "skiing"). Figure 9.1 underscores this issue. One way of improving the quality of the 3D action proposals is to generate anchor micro-tube proposals instead of anchor cuboids. Figure 9.1 **(b)** illustrates the efficacy of the anchor micro-tubes. Note that for a "horse riding" action an anchor micro-tube much improves the spatio-temporal overlap with the ground truth as compared to the anchor cuboid proposal.

9.2.2 Stepping towards an optimal solution to the action detection problem

We plan to move towards using longer anchor micro-tubes (Section 9.2.1) i.e., instead of generating action region hypotheses between pairs of frames, we can generate longer action proposals which span over multiple (n) video frames. One way of generating such proposals is to train an HMM to learn the transition probabilities of frame-level 2D anchor boxes over time. We envision training a network using longer anchor micro-tubes will allow the network to learn spatio-temporal action representations more effectively as they encompass longer temporal extents of action instances. Such a model is expected to provide an optimal solution to the action detection problem (Section 2.2).

9.2.3 Improving action representation

In Chapter 5, we have quantitatively demonstrated that improvements in the deep representation (i.e., using a Faster R-CNN architecture in place of a R-CNN) lead to commensurate improvements in action detection performance. In Chapter 7, we have further improved the action representation by introducing a video-level representation in place of a frame-level once. Further improvements to the current action representation can be made on several fronts. In this work, we have used various deep network architectures composed of 2D convolutional layers which are ideally designed for image processing tasks such as image classification, object detections in images. A promising research direction would be to investigate different deep network architectures specifically designed for video data processing [96, 98, 168, 169]. For instance, very recently Carreira & Zisserman [169] have proposed I3D (inflated 3D) convolutional layers to learn spatio-temporal action representation from videos which have shown state-of-the-art action classification results on all the standard benchmarks (UCF-101, J-HMDB-51) including the largest Kinetics human action video dataset to date.

I3D convolutional layers are explicitly designed to learn spatio-temporal representation from videos as opposed to the classical 2D convolutional layers which can encode only spatial (or appearance) information and suitable for images.

Another interesting direction to improve the action representation is to design a model which is robust to variance in scales i.e., able to detect actions at vastly different scales. In Chapter 5 and 6, we have built our action detection frameworks based on the Faster-RCNN and R-CNN deep network architectures respectively. Whereas, in Chapter 7 and 8, we have extended the Faster-RCNN network architecture for video-based feature representation. Although these deep networks are robust to variations in scale, they learn representation from features computed on a single input scale. Recent advancements in object detection [170, 171] have demonstrated that to achieve the most accurate results we need a model which can explicitly learn representation from features computed on multiple scales. Such representation enables a model to detect actions/objects across vastly different scales.

The inter-frame data-association problem (Section 1.3.4) becomes even more harder when the number of action instances per video frame increases. For instance, let's revisit the "biking" example (see Figure 1.1) in which there are multiple bikers and some of them may carry almost similar appearance and motion throughout the entire sequence. In such cases, solving temporal association is challenging due to several factors such as partial occlusion, position-swapping (Figure 1.7). We have addressed this problem either by using a Viterbi algorithm to link frame-level detections in time (in Chapter 5 & 6) or by learning a video-based action representation combined with a Viterbi algorithm (in Chapter 7 & 8). We believe that more powerful representation can be learnt by incorporating correlation features [172–174] that represent action co-occurrences across time to aid the network to learn better temporal correspondence between inter-frame action regions. For instance, Feichtenhofer *et al.* [174] recently proposed a novel approach to jointly solve object detection and tracking in videos by exploiting both convolutional and cross-correlation features. They have achieved state-of-the-art results on the large-scale ImageNet VID dataset.

9.2.4 Improving temporal action detection

One of the most challenging problems in action detection is the temporal action localisation. Experimental results on UCF-101-24 and LIRIS HARL datasets confirm that there is still a need to improve the temporal detection performance. Many failure cases have been identified due to inaccurate temporal detection. In

particular, for UCF-101 action classes such as "basketball", "cricket bowling", "volleyball spiking" the player appears throughout the entire video sequence whereas the ground truth action is present for a relatively shorter temporal duration compared to the duration of the whole video (Section 4.1). Due to this reason, the frame-level detections for those video frames where the actor is present but not the ground truth action, still have high confidence scores (for that particular action class) leading to inaccurate temporal detection (Section A.5.2). One way of solving this problem may be to train the detector on both foreground (action is present) and background (no action) video frames. Another solution is to learn a separate representation that might be more robust for detecting the start and end of the action. Substantial amount of research has been undertaken in this direction [175–178]. We would like to explore different possibilities to learn better feature representation for accurate temporal detection.

Much work will need to follow. AMTnet-Flow's combination of high accuracy and fast detection speed at test time is very promising for online applications, for instance smart car navigation. As the next step we plan to make our tube generation and labelling algorithm fully incremental and online, by only using region proposals from a subset of video frames at test time and updating the dynamic programming optimisation step at every incoming frame pairs as in [12]. As the search space of 3D proposals is twice the dimension of that for 2D proposals, efficient parallelisation and search are crucial to fully exploit the potential of this approach. Further down the road we wish to extend the idea of micro-tubes to longer time intervals, posing severe challenges in terms of efficient regression in higher-dimensional spaces.

Appendices

Appendix A

- The optical flow map computation details are presented in Section A.1.
- The Region Proposal Network (RPN) training objective is explained in Section A.2.
- The 2D connected component and region proposal generation step is presented in Section A.3.
- The Selective Search region proposal pruning step (used in [14]) is explained in Section A.4.
- The action tube problem formulation is presented in Section A.5.

A.1 Optical flow maps

The optical flow images (or flow maps) have been extensively used throughout this thesis to encode the motion patterns of several human actions in videos. In Chapter 5 & 6, the motion streams (CNNs) take as an input a single flow map, whereas a stack of 5 flow maps are passed as an input to the motion stream in Chapter 8. Recently, Peng *et al.* [32] has demonstrated that the discriminative power of motion stream can be enhanced by stacking flow maps. Flow maps are typically generated by computing dense optical flow between two consecutive video frames. The flow is thus computed using an energy minimisation approach in which a displacement field is solved at different input scales. We have used a popular dense optical flow generation algorithm proposed by Brox *et al.* [158].

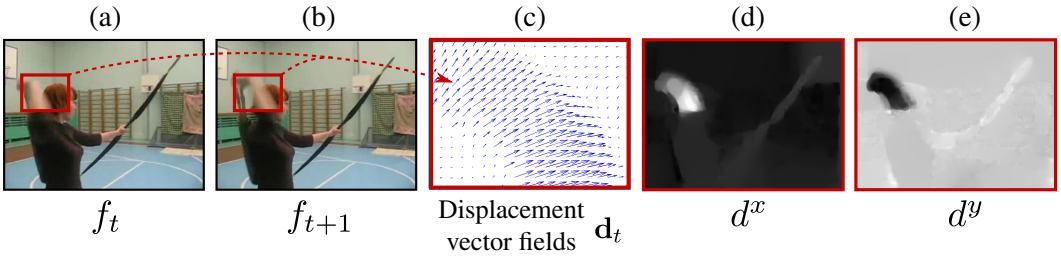


Figure A.1: Dense optical flow computation from a pair of video frames (f_t, f_{t+1}). *The above figure is a slightly modified version of Figure 2 in [27].*

A.1.1 Dense optical flow computation

Consider Figure A.1. A pair of consecutive video frames (f_t, f_{t+1}) containing a "shoot bow" action is shown in **(a)** & **(b)** where the area around a moving hand is depicted by the red rectangular box. A zoomed-in version of the dense optical flow (within the red rectangular box) is shown in **(c)**. Note that the dense optical flow computed between frame f_t and f_{t+1} can be considered as a set of displacement vector fields $\mathbf{d}_t$. A subset of the displacement vector fields is shown in **(c)**.

Figure A.2 illustrates a displacement vector. Consider a pixel P moves from a location (x_t, y_t) (in frame f_t) to its corresponding location (x_{t+1}, y_{t+1}) (in frame f_{t+1}) in time. The displacement of the pixel P can be represented using the displacement vector $\vec{\mathbf{v}}$. The horizontal and vertical components of the displacement vector are denoted as d_t^x and d_t^y.

The horizontal and vertical components of the displacement vector field $\mathbf{d}_t^x$ and $\mathbf{d}_t^y$ for frame f_t and f_{t+1} (Figure A.1 **(a)** & **(b)**) can be seen as the first and second image channels. See **(d)** & **(e)** where $\mathbf{d}_t^x$ and $\mathbf{d}_t^y$ are visualised as

2D grayscale images. Higher intensity denotes positive values, lower intensity denotes negative values. Note that **(d)** & **(e)** highlight the moving hand and bow. Similarly, we can consider the magnitudes of the displacement vector field $\|\mathbf{d}_t\|$ as the third image channel, and then can represent a dense optical flow (between f_t and f_{t+1}) as a 3 channel RGB image which we term as optical flow image/map/heat-map (Figure 2.6).

Implementation note. During implementation, following [14]'s work we first scale up each displacement vector $\vec{\mathbf{v}}$ in the vector field $\mathbf{d}_t$ as follows:

$$\vec{\mathbf{v}} \begin{bmatrix} d_t^x \\ d_t^y \end{bmatrix} = \vec{\mathbf{v}} \begin{bmatrix} d_t^x \\ d_t^y \end{bmatrix} \times 16 + 128 \tag{A.1}$$

and subsequently, set the scaled up values of the horizontal and vertical components d_t^x and d_t^y of each vector $\vec{\mathbf{v}}$:
if $d_t^x < 0$ **then** $d_t^x = 0$; **if** $d_t^x > 255$ **then** $d_t^x = 255$ and
if $d_t^y < 0$ **then** $d_t^y = 0$; **if** $d_t^y > 255$ **then** $d_t^y = 255$.

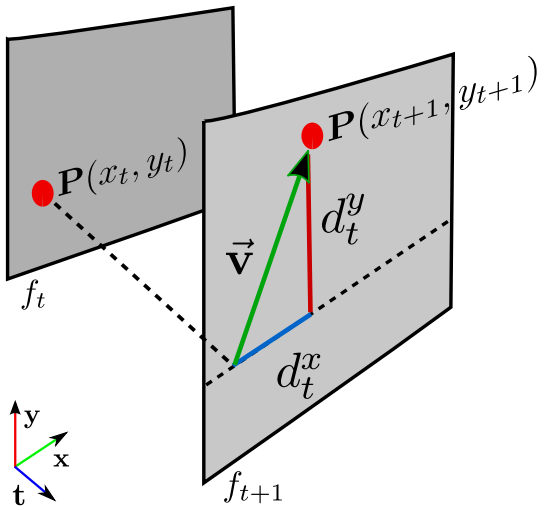


Figure A.2: An illustration of a displacement vector $\vec{\mathbf{v}}$. A pixel P has a location (x_t, y_t) on the 2D image plane at time t. It moves to a new location (x_{t+1}, y_{t+1}) at time $t+1$. The displacement of the pixel can be represented using a vector $\vec{\mathbf{v}}$ and its horizontal and vertical components are denoted as d_t^x and d_t^y respectively.

A.2 RPN training objective

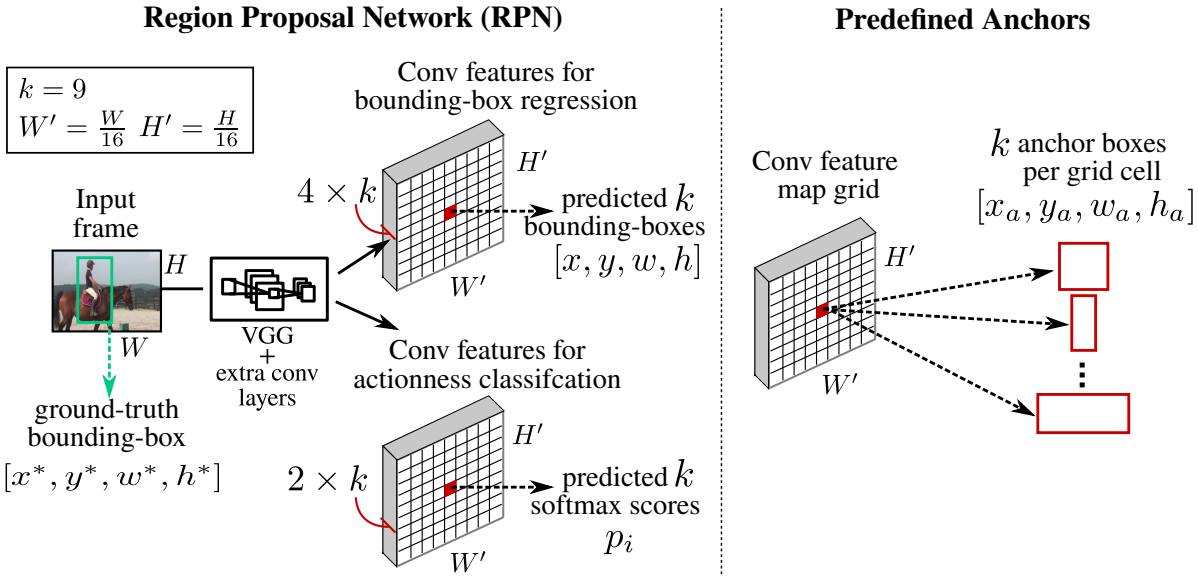


Figure A.3: Region Proposal Network (RPN) [29].

The following is the training objective of an RPN network [29] :

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{A.2}$$

where,

- i index of an anchor in a mini-batch,
- p_i predicted actionness probability of anchor i (action or background),
- p_i^* ground truth label of anchor i (1/0 - positive/negative),
- t_i predicted bounding box,
- t_i^* ground truth box associated with a positive anchor i,
- L_{cls} classifcation loss - cross-entropy loss over 2 classes,
- L_{reg} regression loss - smooth L1 loss,
- N_{cls} mini-batch size (256) and $Nreg$ (no. of anchor locations) normalisation constants.

$$t_x = (x - x_a)/w_a, \qquad t_y = (y - y_a)/h_a,$$

$$
\begin{aligned}
t_w &= log(w/w_a), & t_h &= log(h/h_a), \\
t_x^* &= (x^* - x_a)/w_a, & t_y^* &= (y^* - y_a)/h_a, \\
t_w^* &= log(w^*/w_a), & t_h^* &= log(h^*/h_a).
\end{aligned} \tag{A.3}
$$

- where x, y, w, h denote box's center coordinates and its width and height,
- Variables x, x_a and x^* denote predicted, anchor and ground truth boxes respectively (likewise for y, w, h).

A.3 Region proposal generation using 2D connected components

Let $\mathcal{S}$ represents a subset of pixels in a binary image I_b. Two pixels p and q are said to be connected in $\mathcal{S}$ if there exists a path between them consisting entirely of pixels in $\mathcal{S}$. For any pixel p in $\mathcal{S}$, the set of pixels that are connected to it in $\mathcal{S}$ is called a *connected component*. We call a set of pixels $\mathbf{r}$ in I_b a "region" of the image if $\mathbf{r}$ is a connected set. Two regions, $\mathbf{r}_i$ and $\mathbf{r}_j$ are said to be *adjacent* if their union forms a connected set. Regions that are not adjacent are said to be *disjoint*[1]. Our region proposal generation algorithm first finds all the disjoint segments $\mathbf{r}$ within the binary segmented image I_b produced by the human motion segmentation. Each disjoint region is associated with a minimum bounding box $b_{\mathbf{r}_\mathrm{i}}$. To get an accurate localization window of the human action, our algorithm takes all the possible combination of these bounding boxes $b_{\mathbf{r}_\mathrm{i}}$ to generate N boxes where:

$$N = \sum_{k=2}^{n} \frac{n!}{k!(n-k)!}, \tag{A.4}$$

n is the number of disjoint region and we take k between a range 2 to n.

[1]We consider 8-*adjacency* when referring to regions [179].

A.4 Pruning Selective Search region proposals at test time

In contrast to the work by Gkioxari and Malik [14], we use a smaller motion threshold value to prune Selective Search boxes, to avoid neglecting human activities which exhibit minor body movements exhibited in the LIRIS HARL [128] "typing on keyboard", "telephone conversation" and "discussion" activities. We apply the motion threshold on the 'actionness' scores which we define as:

$$\mu = \frac{\sum_{i \in \mathbf{r}} f_m(i)}{\sum_{j \in \mathcal{I}} f_m(j)} \tag{A.5}$$

where $f_m(.)$ is the function returning the normalised flow magnitude of each pixel of image $\mathcal{I}$, i and j are the pixel indices for region $\mathbf{r}$ and image $\mathcal{I}$. $\sum_{i \in \mathbf{r}} f_m(i)$ is the 'actionness' measure of region $\mathbf{r}$, and the $\sum_{j \in \mathcal{I}} f_m(j)$ is the 'actionness' measure of the whole image. μ is the measure of how much motion or 'actionness' is included inside region $\mathbf{r}$.

Even a motion threshold value of 0.003 prunes, on average, 1,000 boxes, thus significantly reducing computation and resources. In this way activities associated with very small body motion such as LIRIS HARL [128]'s "typing on keyboard", for instance (as opposed to the "running" or "golf" actions of JHMDB21 [18]) can still be detected using dense optical flow [158].

A.5 Problem formulation for action tube generation

We define a video as a sequence of 2D video frames $\mathcal{V} = \{f_1, \ldots, f_t, \ldots, f_T\}$ and an action tube as a sequence of contiguous frame-level detections (bounding-boxes) connected over time without any holes (Figure A.4). The task is to detect multiple concurrent action instances, and thus, we start by identifying detection boxes in each video frame which are highly likely to contain human actions. We denote each detection bounding-box as $\mathbf{b}$ and its class-specific scores as $\mathbf{s}_c$, where c denotes the action category label, $c \in \{1, \ldots, C\}$. Given a set of detections for an entire video, our goal is to identify sequences of detection boxes most likely to compose action tubes.

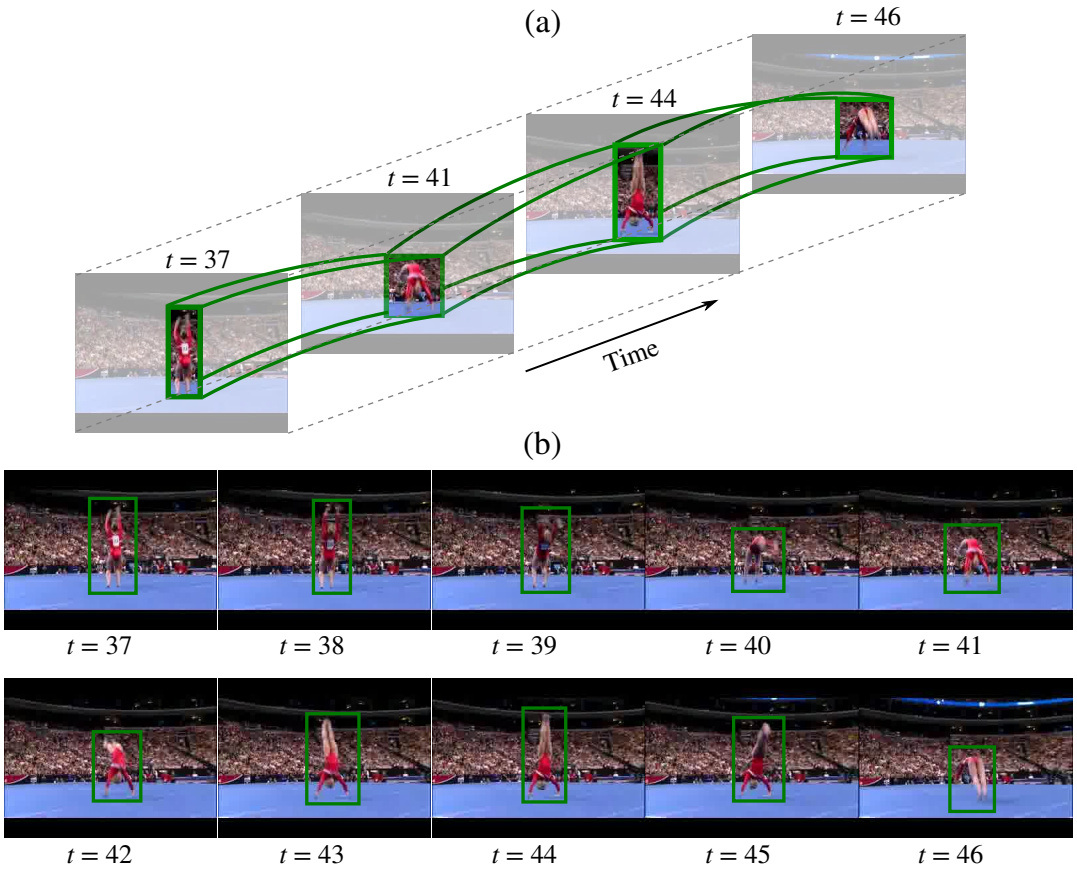


Figure A.4: A "handspring" action is shown in the above video sequence where an acrobat executes a complete revolution of the body. The frame-level detection bounding-boxes are shown in green. ***(a)*** *an action tube (in green) is generated by linking the detections in time which localises the "handspring" action in space and time;* ***(b)*** *the corresponding video frames are shown.*

We cast the action tube extraction as an energy maximisation problem in which configurations of detection boxes in each frame are assigned a cost and

the best action tubes are selected via two passes of dynamic programming. In the first pass, we construct action paths $\mathbf{p}_c$ by associating the detection boxes over time using their class-specific scores as unary and their spatial overlap as pairwise potentials. Candidate action paths $\mathbf{p}_c = [\mathbf{b}_1, \ldots, \mathbf{b}_T]$ initially form a sequence of detection boxes spanning the entire length T of the video. We use a second pass of DP to localise each action in time and to ensure the paths are relatively smooth and have consistent labellings. The final detection results are found by selecting those tubes with the greatest scores.

A.5.1 Constructing action paths

Linking of frame-level detections $\mathbf{b}$ in time is first performed for each action category individually to form action paths. We formulate the temporal linking of detections (i.e. inter-frame data association, Section 1.3.4) into a path finding problem, which will produce K-connected paths for each action class on the whole video, where K is the minimum number of detection boxes extracted in any frame of the video. We can define a temporal data association score (between two detection boxes belonging to two consecutive video frames) to be a sum of unary and pairwise potentials:

$$E_c(\mathbf{b}_t, \mathbf{b}_{t+1}) = \mathbf{s}_c(\mathbf{b}_t) + \mathbf{s}_c(\mathbf{b}_{t+1}) + \lambda \cdot \psi(\mathbf{b}_t, \mathbf{b}_{t+1}), \tag{A.6}$$

where $\psi(\mathbf{b}_t, \mathbf{b}_{t+1})$ is the spatial IoU (Section 4.2.4) of two detection boxes $\mathbf{b}_t$ and $\mathbf{b}_{t+1}$, and λ is a scalar parameter weighting the relative importance of the pairwise term. This energy value of two detection boxes being linked would be high if both boxes have a high score for a particular action class, and if both of them overlap significantly. For each action class we can optimally solve for the action paths by solving:

$$\mathbf{p}_c = \arg\max_{\mathbf{p}} \frac{1}{T} \sum_{t=1}^{T-1} E_c(\mathbf{b}_t, \mathbf{b}_{t+1}) \tag{A.7}$$

where $\mathbf{p}_c = [\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_T]$ is sequence of temporally linked frame-level detection boxes for action class c. We solve the energy maximisation problem (A.7) via dynamic programming.

Once the optimal path has been found, we remove all the detection boxes that form the path and again find another action path until no more paths can be found. For computational efficiency in the subsequent processing steps, we

stop extracting paths after finding the first 3 which have maximum energy. As a result we have multiple paths for each action class in a video. However, human action instances occupy only a fraction of time within the video. Furthermore, instances of the same action class can take place at the same time, and two or more actions instances from different categories may happen concurrently. Therefore, the temporal trimming of the proposed action paths produced by the above procedure is required to achieve action instance detection.

A.5.2 Temporal localisation

Although action paths are associated with individual action classes, because of the way they are constructed (Equation A.7), the scores of frame-level detections $\mathbf{b}_t \in \mathbf{p}_c$ within a path $\mathbf{p}_c$ might not be consistent. Therefore, we formulate the temporal action detection (localisation) as a labeling problem [56]. More specifically, every detection box $\mathbf{b}_t \in \mathbf{p}_c$ must be assigned a label $l_t \in \{1, \ldots, C\}$ that denotes the action ID the detection belongs to. Thus, the goal is to find the paths labelling $\mathbf{L}_{\mathbf{p}_c} = [l_1, \ldots, l_t, \ldots, l_T]$ subject to the constraints that: i) $\mathbf{L}_{\mathbf{p}_c}$ should be consistent with the observations, ii) $\mathbf{L}_{\mathbf{p}_c}$ is piece-wise constant i.e. it is smooth in order to avoid sudden jumps in labels assigned to detections belong to consecutive frames. Note that, the paths labelling addresses the problem of temporal action detection i.e. the piece-wise constant temporal segments help to identify the start and end points of action instances present in a video.

The paths labelling problem can be cast into an energy maximisation framework, that is, the labelling $\mathbf{L}_{\mathbf{p}_c}$ that maximises the energy can be estimated using:

$$E(\mathbf{L}_{\mathbf{p}_c}) = E_D(\mathbf{L}_{\mathbf{p}_c}) - E_S(\mathbf{L}_{\mathbf{p}_c}) \tag{A.8}$$

where $E_D(\mathbf{L}_{\mathbf{p}_c})$ (the data term) measures the similarity between $\mathbf{L}_{\mathbf{p}_c}$ and the observations, and $E_S(\mathbf{L}_{\mathbf{p}_c})$ (the smoothness term) penalises labelling that are not piece-wise constant (or cases where there are label jumps).

The data-term is typically defined as:

$$E_D(\mathbf{L}_{\mathbf{p}_c}) = \sum_{t=1}^{T} s_{l_t}(\mathbf{b}_t), \tag{A.9}$$

where, $s_{l_t}(\mathbf{b}_t)$ measures how appropriate the label l_t is for t-th detection box $\mathbf{b}_t$. To evaluate this appropriateness, we use a softmax layer (of a deep neural network) in Chapter 5 and a multi-class SVM classifier in Chapter 6. More specifically, $s_{l_t}(\mathbf{b}_t)$ denotes the probability score of assigning the label l_t to a

detection bounding-box $\mathbf{b}_t \in \mathbf{p}_c$.

To achieve smooth labelling of action paths, the smoothness term $E_S(\mathbf{L}_{\mathbf{p}_c})$ makes sure that the pairs of successive detection boxes $(\mathbf{b}_t, \mathbf{b}_{t+\Delta})$ (belonging to an action path $\mathbf{p}_c$) are assigned a same class label c. It does so by penalsing the assignment of different labels to successive detections. if we consider that the detections belonging to consecutive video frames (f_t, f_{t+1}), i.e. under a first-order Markovian assumption, the term can be written as a summation of pairwise potentials, namely:

$$E_S(\mathbf{L}_{\mathbf{p}_c}) = \sum_{t=1}^{T-1} V(l_t, l_{t+1}). \tag{A.10}$$

The piece-wise constant labelling constraint is enforced by the following potential function:

$$V(l_t, l_{t+1}) = \begin{cases} 0 & \text{if } l_t = l_{t+1} \\ \alpha & \text{otherwise,} \end{cases} \tag{A.11}$$

where, α is a constant term and we set the value of: (a) α by cross validating it separately for each action category on the training sets in Chapter 5, and (b) $\alpha = 3$ for all action classes from cross validation on the training set of LIRIS HARL dataset in Chapter 6. In order to efficiently solve the global optimisation problem we use a dynamic programming approach.

Let's assume that the dynamic programming (DP) matrix M is of shape $|C| \times (T+1)$, if $M_t[l_t]$ denotes the maximum labelling cost for the first t frames provided that the t-th frame has label l_t, the following recursive equation (t is increasing) fills in the matrix M:

$$M_1[l_1] = s_{l_1}(\mathbf{b}_1) \tag{A.12}$$

$$M_t[l_t] = s_{l_t}(\mathbf{b}_t) + \max_{l_{t-1}}(M_{t-1}(l_{t-1}) - V(l_{t-1}, l_t)). \tag{A.13}$$

In Eq. A.12, $s_{l_1}(\mathbf{b}_1)$ denotes the probability score of assigning the label l to a detection bounding-box $\mathbf{b}$ at time $t = 1$ where $\mathbf{b} \in \mathbf{p}_c$ and $l \in \{1, 2, \ldots, C\}$. Once the DP matrix M is completely filled in for all frames in a video, the optimal solution can be found by first seeking the label of the last frame, i.e.,

$$l_T^* = \operatorname*{argmax}_{l_T} M_T[l_T], \tag{A.14}$$

and then by tracking back in order of decreasing t:

$$l_t^* = \underset{l_i}{\text{argmax}} \ (M_t[l_t] - V(l_{t-1}, l_t^*)), \tag{A.15}$$

This will give us a piece-wise constant label for each detection box $\mathbf{b}_t$ of the action path. The labelling of this path can be written as $\mathbf{L}_{\mathbf{p}_c} = [l_1, l_2, l_3, \ldots l_T]$.

References

[1] B. Romera-Paredes and P. H. S. Torr, "Recurrent instance segmentation," in *European Conference on Computer Vision*, pp. 312–329, Springer, 2016.

[2] I. Laptev and P. Pérez, "Retrieving actions in movies," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8, IEEE, 2007.

[3] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 3384–3391, IEEE, 2010.

[4] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2556–2563, IEEE, 2011.

[5] R. Navarathna, P. Lucey, P. Carr, E. Carter, S. Sridharan, and I. Matthews, "Predicting movie ratings from audience behaviors," in *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pp. 1058–1065, IEEE, 2014.

[6] Z. Deng, R. Navarathna, P. Carr, S. Mandt, Y. Yue, I. Matthews, and G. Mori, "Factorized variational autoencoders for modeling audience reactions to movies," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2577–2586, 2017.

[7] K. Korosec, "Toyota is betting on this startup to drive its self-driving car plans forward." Available at: `http://fortune.com/2017/09/27/toyota-self-driving-car-luminar/`.

[8] G. Inc., "Self-driving car project - google." Available at: `https://waymo.com/tech/`.

[9] F. Cuzzolin, "SARAS - Smart Autonomous Robotic Assistant Surgeon." Available at: http://cms.brookes.ac.uk/staff/FabioCuzzolin/.

[10] en.wikipedia.org, "Robot-assisted surgery." Available at: https://en.wikipedia.org/wiki/Robot-assisted_surgery.

[11] allaboutroboticsurgery.com, "Recognising and localising human actions, ph.d. dissertation submitted at oxford brookes university." Available at: http://allaboutroboticsurgery.com/surgicalrobots.html.

[12] G. Singh, S. Saha, M. Sapienza, P. Torr, and F. Cuzzolin, "Online real-time multiple spatiotemporal action localisation and prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3637–3646, 2017.

[13] M. Sapienza, "Recognising and localising human actions." Available at: https://radar.brookes.ac.uk/radar/items/8c520676-aef0-4b67-a160-b6dd8e6a3e58/1/.

[14] G. Gkioxari and J. Malik, "Finding action tubes," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2015.

[15] A. Bronstein, M. Bronstein, and R. Kimmel, "Topology-invariant similarity of nonrigid shapes," *Int. Journal of Computer Vision*, vol. 81, no. 3, pp. 281–301, 2009.

[16] R. Poppe, "A survey on vision-based human action recognition," *Image and Vision Computing*, vol. 28, pp. 976–990, 2010.

[17] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human action classes from videos in the wild," tech. rep., CRCV-TR-12-01, 2012.

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

[19] A. Milan, S. H. Rezatofighi, A. R. Dick, I. D. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks.," in *AAAI*, pp. 4225–4232, 2017.

[20] P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Learning to track for spatio-temporal action localization," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, June 2015.

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

[22] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, and C. G. Snoek, "Action localization with tubelets from motion," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 740–747, 2014.

[23] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid, "Spatio-temporal object detection proposals," in *European conference on computer vision*, pp. 737–752, Springer, 2014.

[24] J. C. van Gemert, M. Jain, E. Gati, C. G. Snoek, *et al.*, "Apt: Action localization proposals from dense trajectories," 2015.

[25] M. Marian Puscas, E. Sangineto, D. Culibrk, and N. Sebe, "Unsupervised tube extraction using transductive learning and dense trajectories," in *Proceedings of the IEEE international conference on computer vision*, pp. 1653–1661, 2015.

[26] G. Yu and J. Yuan, "Fast action proposals for human action detection and search," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1302–1311, 2015.

[27] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, pp. 568–576, 2014.

[28] R. Girshick, J. Donahue, T. Darrel, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2014.

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

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

[31] J. Lu, r. Xu, and J. J. Corso, "Human action segmentation with hierarchical supervoxel consistency," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, June 2015.

[32] X. Peng and C. Schmid, "Multi-region two-stream r-cnn for action detection," in *European Conference on Computer Vision*, pp. 744–759, Springer, 2016.

[33] 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 *British Machine Vision Conference*, 2016.

[34] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[35] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, pp. 2017–2025, 2015.

[36] J. Johnson, A. Karpathy, and L. Fei-Fei, "Densecap: Fully convolutional localization networks for dense captioning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[37] S. Saha, G. Singh, M. Sapienza, P. H. Torr, and F. Cuzzolin, "Spatio-temporal human action localisation and instance segmentation in temporally untrimmed videos," *arXiv preprint arXiv:1707.07213*, 2017.

[38] S. Saha, G. Singh, and F. Cuzzolin, "Amtnet: Action-micro-tube regression by end-to-end trainable deep architecture," in *Proc. Int. Conf. Computer Vision*, 2017.

[39] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 3, pp. 500–513, 2011.

[40] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient hierarchical graph-based video segmentation," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2141–2148, IEEE, 2010.

[41] J. Lu, J. J. Corso, *et al.*, "Human action segmentation with hierarchical supervoxel consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3762–3771, 2015.

[42] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

[43] 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.

[44] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *null*, p. 1470, IEEE, 2003.

[45] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," *Computer Vision–ECCV 2010*, pp. 143–156, 2010.

[46] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[47] 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.

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *European Conference on Computer Vision*, pp. 346–361, Springer, 2014.

[49] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, 2015.

[50] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *European Conference on Computer Vision*, pp. 391–405, Springer, 2014.

[51] W. Chen, C. Xiong, R. Xu, and J. J. Corso, "Actionness ranking with lattice conditional ordinal random fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 748–755, 2014.

[52] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[53] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid, "Action tubelet detector for spatio-temporal action localization," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

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

[55] K. Soomro, H. Idrees, and M. Shah, "Action localization in videos through context walk," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

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

[57] A. Arnab and P. H. Torr, "Pixelwise instance segmentation with a dynamically instantiated network," *arXiv preprint arXiv:1704.02386*, 2017.

[58] Z. Zhang, S. Fidler, and R. Urtasun, "Instance-level segmentation for autonomous driving with deep densely connected mrfs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 669–677, 2016.

[59] M. A. Goodale and A. D. Milner, "Separate visual pathways for perception and action," *Trends in neurosciences*, vol. 15, no. 1, pp. 20–25, 1992.

[60] J. K. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, p. 16, 2011.

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

[62] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer vision and image understanding*, vol. 115, no. 2, pp. 224–241, 2011.

[63] S. Herath, M. Harandi, and F. Porikli, "Going deeper into action recognition: A survey," *Image and Vision Computing*, vol. 60, pp. 4–21, 2017.

[64] C. Schuldt, 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.

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

[66] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pp. 65–72, IEEE, 2005.

[67] G. Willems, T. Tuytelaars, and L. Van Gool, "An efficient dense and scale-invariant spatio-temporal interest point detector," *Computer Vision–ECCV 2008*, pp. 650–663, 2008.

[68] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th ACM international conference on Multimedia*, pp. 357–360, ACM, 2007.

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

[70] L. Yeffet and L. Wolf, "Local trinary patterns for human action recognition," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 492–497, IEEE, 2009.

[71] A. Klaser, M. Marszałek, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *BMVC 2008-19th British Machine Vision Conference*, pp. 275–1, British Machine Vision Association, 2008.

[72] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8, Ieee, 2007.

[73] C. Harris and M. Stephens, "A combined corner and edge detector.," in *Alvey vision conference*, vol. 15, pp. 10–5244, Manchester, UK, 1988.

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

[75] P. Sand and S. Teller, "Particle video: Long-range motion estimation using point trajectories," *International Journal of Computer Vision*, vol. 80, no. 1, p. 72, 2008.

[76] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," *Computer Vision–ECCV 2010*, pp. 282–295, 2010.

[77] J. Lezama, K. Alahari, J. Sivic, and I. Laptev, "Track to the future: Spatio-temporal video segmentation with long-range motion cues," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 3369–3376, IEEE, 2011.

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

[79] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International journal of computer vision*, vol. 103, no. 1, pp. 60–79, 2013.

[80] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.

[81] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," *Image analysis*, pp. 363–370, 2003.

[82] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *European conference on computer vision*, pp. 428–441, Springer, 2006.

[83] H. Uemura, S. Ishikawa, and K. Mikolajczyk, "Feature tracking and motion compensation for action recognition.," in *BMVC*, pp. 1–10, 2008.

[84] H. Wang, D. Oneata, J. Verbeek, and C. Schmid, "A robust and efficient video representation for action recognition," *International Journal of Computer Vision*, vol. 119, no. 3, pp. 219–238, 2016.

[85] M. Jain, H. Jegou, and P. Bouthemy, "Better exploiting motion for better action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2555–2562, 2013.

[86] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer vision–ECCV 2006*, pp. 404–417, 2006.

[87] A. A. Efros, A. C. Berg, G. Mori, and J. Malik, "Recognizing action at a distance," in *null*, p. 726, IEEE, 2003.

[88] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, pp. 1395–1402, IEEE, 2005.

[89] M. Raptis, I. Kokkinos, and S. Soatto, "Discovering discriminative action parts from mid-level video representations," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1242–1249, IEEE, 2012.

[90] Y. Wang and G. Mori, "Hidden part models for human action recognition: Probabilistic versus max margin," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1310–1323, 2011.

[91] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[92] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[93] 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.

[94] 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.

[95] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *International Conference on Learning Representations*, 2015.

[96] 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.

[97] 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.

[98] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497, 2015.

[99] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[100] Y. Li, W. Li, V. Mahadevan, and N. Vasconcelos, "VLAD3: Encoding dynamics of deep features for action recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[101] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao, "A key volume mining deep framework for action recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[102] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould, "Dynamic image networks for action recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[103] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang, "Real-time action recognition with enhanced motion vector CNNs," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[104] R. Arandjelovic and A. Zisserman, "All about VLAD," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1578–1585, 2013.

[105] 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.

[106] A. Gorban, H. Idrees, Y. Jiang, A. R. Zamir, I. Laptev, M. Shah, and R. Sukthankar, "Thumos challenge: Action recognition with a large number of classes," in *CVPR workshop*, 2015.

[107] S. Escalera, X. Baró, J. Gonzalez, M. A. Bautista, M. Madadi, M. Reyes, V. Ponce-López, H. J. Escalante, J. Shotton, and I. Guyon, "Chalearn looking at people challenge 2014: Dataset and results," in *Computer Vision-ECCV 2014 Workshops*, pp. 459–473, Springer, 2014.

[108] 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," *arXiv preprint arXiv:1507.05738*, 2015.

[109] J. Yuan, Z. Liu, and Y. Wu, "Discriminative subvolume search for efficient action detection," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 2442–2449, IEEE, 2009.

[110] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce, "Automatic annotation of human actions in video," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1491–1498, IEEE, 2009.

[111] A. Gaidon, Z. Harchaoui, and C. Schmid, "Temporal localization of actions with actoms," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2782–2795, 2013.

[112] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments for activity classification," in *European conference on computer vision*, pp. 392–405, Springer, 2010.

[113] D. Oneata, J. Verbeek, and C. Schmid, "Efficient action localization with approximately normalized fisher vectors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2545–2552, 2014.

[114] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Efficient subwindow search: A branch and bound framework for object localization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 12, pp. 2129–2142, 2009.

[115] A. Richard and J. Gall, "Temporal action detection using a statistical language model," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[116] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1764–1772, 2014.

[117] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[118] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *SSST Workshop*, 2014.

[119] Z. Shou, D. Wang, and S.-F. Chang, "Temporal action localization in untrimmed videos via multi-stage cnns," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

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

[121] J. Yuan, B. Ni, X. Yang, and A. A. Kassim, "Temporal action localization with pyramid of score distribution features," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[122] L. Cao, Z. Liu, and T. S. Huang, "Cross-dataset action detection," in *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pp. 1998–2005, IEEE, 2010.

[123] D. Tran, J. Yuan, and D. Forsyth, "Video event detection: From subvolume localization to spatiotemporal path search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 2, pp. 404–416, 2014.

[124] Y. Tian, R. Sukthankar, and M. Shah, "Spatiotemporal deformable part models for action detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2642–2649, 2013.

[125] A. Prest, V. Ferrari, and C. Schmid, "Explicit modeling of human-object interactions in realistic videos," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 4, pp. 835–848, 2013.

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

[127] 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.

[128] 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., LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon, 2012.

[129] L. Wang, Y. Qiao, and X. Tang, "Video action detection with relational dynamic-poselets," in *European Conference on Computer Vision*, pp. 565–580, Springer, 2014.

[130] S. Manen, M. Guillaumin, and L. Van Gool, "Prime object proposals with randomized prim's algorithm," in *Proceedings of the IEEE international conference on computer vision*, pp. 2536–2543, 2013.

[131] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proceedings of the IEEE international conference on computer vision*, pp. 3551–3558, 2013.

[132] S. D. Jain and K. Grauman, "Supervoxel-consistent foreground propagation in video," in *European Conference on Computer Vision*, pp. 656–671, Springer, 2014.

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

[134] L. Wang, Y. Qiao, X. Tang, and L. Van Gool, "Actionness estimation using hybrid fully convolutional networks," *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2016.

[135] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[136] J. Winn and J. Shotton, "The layout consistent random field for recognizing and segmenting partially occluded objects," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 37–44, IEEE, 2006.

[137] Y. Yang, S. Hallman, D. Ramanan, and C. C. Fowlkes, "Layered object models for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1731–1743, 2012.

[138] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *European Conference on Computer Vision*, pp. 297–312, Springer, 2014.

[139] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 447–456, 2015.

[140] Y.-T. Chen, X. Liu, and M.-H. Yang, "Multi-instance object segmentation with occlusion handling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3470–3478, 2015.

[141] K. Li, B. Hariharan, and J. Malik, "Iterative instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3659–3667, 2016.

[142] 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 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–970, 2015.

[143] 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`.

[144] 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`.

[145] M. Dillon, "Introduction to modern information retrieval: G. salton and m. mcgill. mcgraw-hill, new york (1983).," 1983.

[146] D. M. Christopher, R. Prabhakar, and S. Hinrich, "Introduction to information retrieval," *An Introduction To Information Retrieval*, vol. 151, no. 177, p. 5, 2008.

[147] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[148] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

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

[150] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "Pascal VOC development kit code." Available at: `http://host.robots.ox.ac.uk/pascal/VOC/voc2012/#devkit`.

[151] 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.

[152] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *arXiv preprint arXiv:1506.02640*, 2015.

[153] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[154] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results." Available at: `http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html`.

[155] C. Xu, C. Xiong, and J. J. Corso, "Streaming hierarchical video segmentation," in *European Conference on Computer Vision*, pp. 626–639, Springer, 2012.

[156] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[157] P. Kohli, P. H. Torr, *et al.*, "Robust higher order potentials for enforcing label consistency," *International Journal of Computer Vision*, vol. 82, no. 3, pp. 302–324, 2009.

[158] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," *Computer Vision-ECCV 2004*, pp. 25–36, 2004.

[159] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *CoRR*, 2013.

[160] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *CoRR*, vol. abs/1408.5093, 2014.

[161] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[162] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[163] S. Saha, G. Singh, M. Sapienza, P. H. S. Torr, and F. Cuzzolin, "Supplementary material: Deep learning for detecting multiple space-time action tubes in videos," in *British Machine Vision Conference*, 2016. `http://tinyurl.com/map6lde`.

[164] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, no. EPFL-CONF-192376, 2011.

[165] jcjohnson, "densecap," 2016. `https://github.com/jcjohnson/densecap`.

[166] qassemoquab, "stnbhwd," 2015. `https://github.com/qassemoquab/stnbhwd`.

[167] M. Pall, "The luajit project," *Web site: http://luajit. org*, 2008.

[168] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *European conference on computer vision*, pp. 140–153, Springer, 2010.

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

[170] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[171] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, vol. 1, p. 4, 2017.

[172] 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.

[173] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*, pp. 850–865, Springer, 2016.

[174] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3038–3046, 2017.

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

[176] I. C. Duta, B. Ionescu, K. Aizawa, N. Sebe, *et al.*, "Spatio-temporal vector of locally max pooled features for action recognition in videos," in *30TH IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR 2017)*, pp. 3205–3214, IEEE, 2017.

[177] 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 *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1417–1426, IEEE, 2017.

[178] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. C. Niebles, "Sst: Single-stream temporal action proposals," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 6373–6382, IEEE, 2017.

[179] R. C. Gonzalez and R. E. Woods, “Digital image processing,” New Jersey 07458: Pearson Prentice Hall, 2008.