

# Supplementary Material: Deep Learning for Detecting Multiple Space-Time Action Tubes in Videos

Suman Saha<sup>1</sup>  
suman.saha-2014@brookes.ac.uk

Gurkirt Singh<sup>1</sup>  
gurkirt.singh-2015@brookes.ac.uk

Michael Sapienza<sup>2</sup>  
michael.sapienza@eng.ox.ac.uk

Philip H. S. Torr<sup>2</sup>  
philip.torr@eng.ox.ac.uk

Fabio Cuzzolin<sup>1</sup>  
fabio.cuzzolin@brookes.ac.uk

<sup>1</sup> Dept. of Computing and  
Communication Technologies  
Oxford Brookes University  
Oxford, UK

<sup>2</sup> Department of Engineering Science  
University of Oxford  
Oxford, UK

---

## 1 Implementation details

**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 [3, 4].

**Optical flow based video frame generation.** We computed dense optical flow between each pair of consecutive video frames using the state-of-the-art algorithm in [5]. The 3-channel optical flow values (i.e., flow- $x$ , flow- $y$  and the flow magnitude) were then used to construct ‘motion frames’ [2]. These motion (flow) frames were used to train the motion-based RPN and Fast R-CNN networks.

**Modifications in the existing codebase.** We downloaded the publicly available Faster R-CNN MATLAB code from [https://github.com/ShaoqingRen/faster\\_rcnn](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.

**CNN weight initialisation.** The RPN and Fast R-CNN networks [4] were initialised with weights from a pre-trained ImageNet model [5].

**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.

**Selective Search region proposals.** In Section 3.1 and 3.4, 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 [4] and retain on average 100 SS boxes per frame.

## 2 Evaluation metrics

To compare our results on UCF-101 and J-HMDB-21 with the state of the art, we used the same evaluation metric proposed by [6]. For LIRIS HARL, we used the evaluation tool provided by the LIRIS HARL competition [2]. Note that the Area Under the Curve (AUC) on J-HMDB-21 reported by [2] is sensitive to negative detections, as AUC increases when adding many easy negatives<sup>1</sup> [9], whereas mAP is not affected by easy negatives. The LIRIS HARL evaluation metric [2] 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. The reported metric on UCF-101 and J-HMDB-21 is the mean Average Precision (mAP) at a threshold  $\delta = .2$  for spatio-temporal localisation (UCF-101) and  $\delta = .5$  for spatial localisation (J-HMDB-21). We report an mAP and Integrated F1-Score [2] for the LIRIS-HARL dataset at a threshold of  $\delta = .1$ .

## 3 Experimental results

### 3.1 Comparative analysis of region proposal quality

Firstly we analysed the quality of Selective Search vs RPN-based region proposals using the Recall-to-IoU measure [4]. We extracted 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 applied 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 computed 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 Fig. 1.

### 3.2 Ablation study

We are the first to report an ablation study of the spatio-temporal action localisation performance on UCF-101 dataset. Table 1 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: *HorseRiding*(HR) and *TennisSwing*(TS), while the motion-based detection net outperforms for action classes: *CricketBowling*(CB), *LongJump*(LJ), *SalsaSpin*(SaS) and *SoccerJuggling* (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.

---

<sup>1</sup>Negatives which have lower detection confidence than all positives.

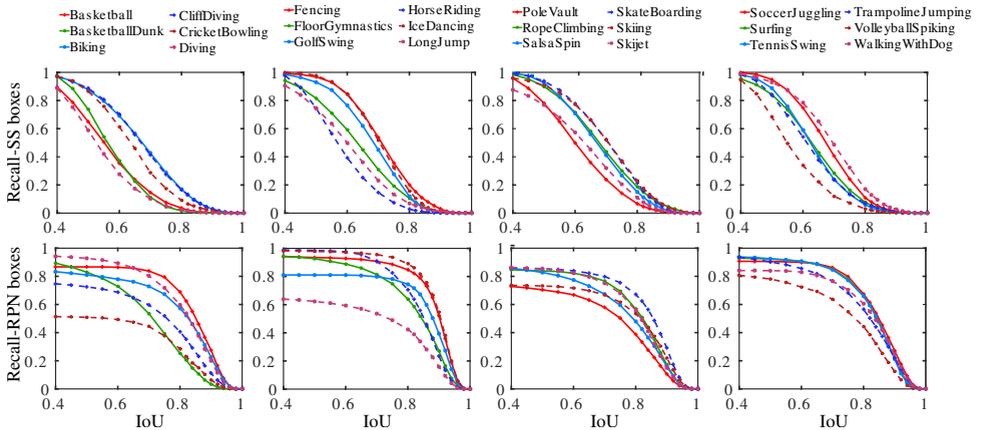


Figure 1: Performance comparison between Selective Search (SS) and RPN-based region proposals on four groups of action classes (vertical columns) in UCF-101. Top row: recall vs. IoU curve for SS. Bottom row: results for RPN-based region proposals.

Table 1: An ablation study of the spatio-temporal detection results (video APs in %) on UCF-101.

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

### 3.3 Impact of label smoothing on detection performance

We conducted experiments to show the significance of the path label smoothing step. More specifically, we show that the class-specific  $\alpha_c$  values help to smooth the action paths for each action category independently resulting 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  $\alpha_c = 0$  for each action category. Then, we use the cross validated class-specific  $\alpha_c$  values and again generated detection results. In our experiment, we set the spatio-temporal IoU threshold  $\delta = 0.2$ . Table 2 presents the results for both the cases: detection result obtained by setting  $\alpha_c = 0$  for each action and result generated using the cross validated class-specific  $\alpha_c$  values. Table 3 shows the class-specific  $\alpha_c$  values obtained by cross validation. Notice that the class-specific  $\alpha_c$  improves the detection accuracy (mAP) by 6%. We empirically observed 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  $\alpha_c$  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, different alpha values for different action

classes is better than having a single alpha value assigned for all classes.

Table 2: Spatio-temporal detection results (mAP) on UCF-101 using two different sets of  $\alpha_c$  values.

	mAP
$\alpha_c = 0$	60.77
class-specific $\alpha_c$	<b>66.75</b>

Table 3: Class specific  $\alpha_c$  values for each action category in UCF-101 obtained from cross validation.

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

### 3.4 Training and test-time detection speed comparison

We also performed an analysis of training and testing time requirements of our method in comparison with our main competitors [2, 5]. Note that [5] modifies the pipeline of ActionTube [2] by adding a ‘tracking by detection’ module - thus in Table 4 and 5, while comparing the computation time, we only consider those components of the detection pipelines which are common to both [2] and [5].

**Comparison on UCF-101 dataset.** The 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  $2\times$  faster in training and  $20\times$  faster in testing on UCF101 (refer Table 4). The most time consuming step in [2, 5] 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 [2]’s approach, for UCF-101’s 240k training video frames with 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 [2, 5] which use an expensive multi-stage classification strategy. We report comparison results on J-HMDB-21 dataset. We exclude our 2<sup>nd</sup> 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

Table 4: Training and test time detection speed comparison on UCF-101 with [2, 6].

<b>Training time:</b> time computed on 2293 UCF-101 training video clips (split-1)			
ActionTube [2] and STMH [6]		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
<b>Total training time required</b>	<b>6+ days</b>	<b>Total training time required</b>	<b>3+ days</b>
<b>Test time:</b> 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 <sup>st</sup> pass DP	76 minutes
		2 <sup>nd</sup> pass DP	7 minutes
<b>Total test time required</b>	<b>2+ days</b>	<b>Total test time required</b>	<b>2.5 hours</b>

steps are common to [2] 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 [2, 6] with our RPN and detection nets computation times. Table 5 shows the time required for each step. The reported computation time is averaged over all the videos in the J-HMDB-21 test split1. The time is in second per video clip. 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  $10\times$  faster than [2] and  $5\times$  than [6] in detecting actions in a video.

Table 5: Test time detection speed comparison on J-HMDB-21 with [2, 6].

ActionTube [2], STMH [6]	Average time (Sec./video)	Ours	Average time(Sec./video)
Selective Search [2] / EdgeBoxes [6]	68.10 / 6.81	RPN proposal generation	4.08
CNN feature extraction	45.42	Detection network	6.81
<b>Avg. detection time</b>	<b>113.52 [2] / 52.23 [6]</b>	<b>Avg. detection time</b>	<b>10.89</b>

### 3.5 Additional spatial and temporal localisation results

Figure 2 provides additional evidence on the temporal detection and spatial localisation performance of our method. The additional results can be seen in the video submitted alongside this document.



Figure 2: Sample 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.

## References

- [1] T Brox, A Bruhn, N Papenbergh, and J Weickert. High accuracy optical flow estimation based on a theory for warping. *Proc. European Conf. Computer Vision*, 2004.
- [2] G Gkioxari and J Malik. Finding action tubes. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2015.
- [3] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. Black. Towards understanding action recognition. *Proc. Int. Conf. Computer Vision*, 2013.
- [4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.
- [5] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.
- [6] Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Learning to track for spatio-temporal action localization. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, June 2015.
- [7] 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*, 127:14–30, 2014.