

# 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

---

## Abstract

In this work, we propose an approach to the spatiotemporal localisation (detection) and classification of multiple concurrent actions within temporally untrimmed videos. Our framework is composed of three stages. In stage 1, appearance and motion detection networks are employed to localise and score actions from colour images and optical flow. In stage 2, the appearance network detections are boosted by combining them with the motion detection scores, in proportion to their respective spatial overlap. In stage 3, sequences of detection boxes most likely to be associated with a single action instance, called action tubes, are constructed by solving two energy maximisation problems via dynamic programming. While in the first pass, action paths spanning the whole video are built by linking detection boxes over time using their class-specific scores and their spatial overlap, in the second pass, temporal trimming is performed by ensuring label consistency for all constituting detection boxes. We demonstrate the performance of our algorithm on the challenging UCF101, J-HMDB-21 and LIRIS-HARL datasets, achieving new state-of-the-art results across the board and significantly increasing detection speed at test time.

## 1 Introduction

Recent advances in object detection via convolutional neural networks (CNNs) [8] have triggered a significant performance improvement in the state-of-the-art action detectors [8, 24]. However, the accuracy of these approaches is limited by their relying on unsupervised region proposal algorithms such as Selective Search [8] or EdgeBoxes [24] which, besides being resource-demanding, cannot be trained for a specific detection task and are disconnected from the overall classification objective. Moreover, these approaches are computationally expensive as they follow a multi-stage classification strategy which requires CNN fine-tuning and intensive feature extraction (at both training and test time), the caching of these features

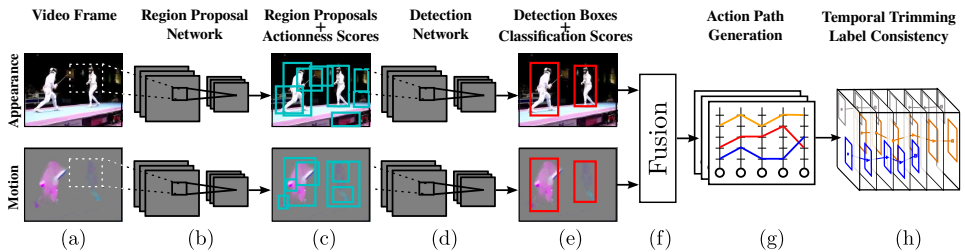


Figure 1: At test time, (a) RGB and optical-flow images are passed to (b) two separate region proposal networks (RPNs). (c) Each network outputs region proposals with associated actionness scores (§ 3.1). (d) Each appearance/motion detection network takes as input the relevant image and RPN-generated region proposals, and (e) outputs detection boxes and softmax probability scores (§ 3.2). (f) Appearance and motion based detections are fused (§ 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 (§ 3.4).

onto disk, and finally the training of a battery of one-vs-all SVMs for action classification. On large datasets such as UCF-101 [25], overall training and feature extraction takes a week using 7 Nvidia Titan X GPUs, 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.

To overcome these issues 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 [20], in which a single CNN is trained for both detecting and classifying frame-level region proposals in an end-to-end fashion. Detected frame-level proposals are subsequently linked in time to form space-time ‘action tubes’ [8] by solving two optimisation problems via dynamic programming. We demonstrate that the proposed action detection pipeline is at least  $2\times$  faster in training and  $5\times$  faster in test time detection speeds as compared to [8, 24]. In the supplementary material, we present a comparative analysis of the training and testing time requirements of our approach with respect to [8, 24] on the UCF-101 [25] and J-HMDB-21 [12] datasets. Moreover, our pipeline consistently outperforms previous state-of-the-art results (§ 4).

**Overview of the approach.** Our approach is summarised in Fig. 1. We train two pairs of Region Proposal Networks (RPN) [14] and Fast R-CNN [6] detection networks - one on RGB and another on optical-flow images [8]. For each pipeline, the RPN (b), takes as input a video frame (a), and generates a set of region proposals (c), and their associated ‘actionness’ [8] scores<sup>1</sup>. Next, a Fast R-CNN [14] detection network (d) takes as input the original video frame and a subset of the region proposals generated by the RPN, and outputs a ‘regressed’ detection box and a softmax classification score for each input proposal, indicating the probability of an action class being present within the box. To merge appearance and motion cues, we fuse (f) the softmax scores from the appearance- and motion-based detection boxes (e) (§ 3.3). 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 [8]). An additional second pass of

<sup>1</sup>A softmax score for a region proposal containing an action or not.

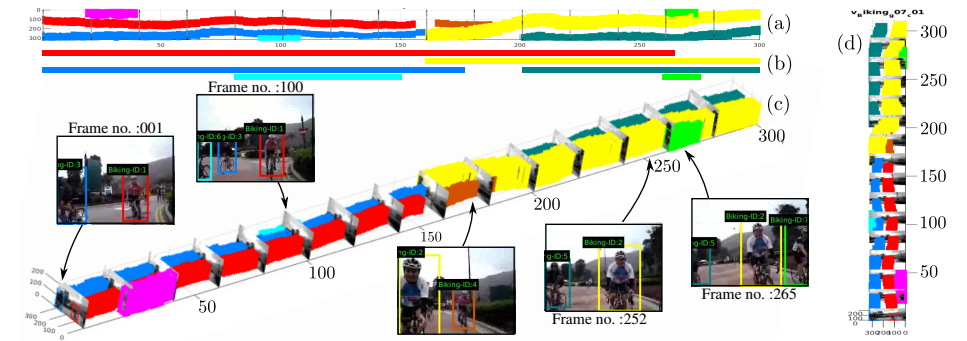


Figure 2: Action tube detection in a ‘biking’ video taken from UCF-101 [24]. (a) Side view of the detected action tubes where each colour represents a particular instance. The detection boxes in each frame are linked up to form space-time action tubes. (b) Illustration of the ground-truth temporal duration 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.

dynamic programming is introduced to take care of temporal detection (h). As a result, our action tubes are not constrained to span the entire video duration, as in [8]. Furthermore, extracting multiple paths allows our algorithm to account for multiple co-occurring instances of the same action class (see Fig. 2).

Although it makes use of existing RPN [24] and Fast R-CNN [6] architectures, this work proposes a radically new approach to spatiotemporal action detection which brings them together with a novel late fusion approach and an original action tube generation mechanism to dramatically improve accuracy and detection speed. Unlike [8, 54], in which appearance and motion information are fused by combining fc7 features, we follow a late fusion approach [24]. 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 (§ 3.3). The 2<sup>nd</sup> pass of dynamic programming, we introduce for action tube temporal trimming, contributes to a great extent to significantly improve the detection performance (§ 4).

**Contributions.** In summary, this work’s main contribution is a novel action detection pipeline which:

- incorporates recent deep Convolutional Neural Network architectures for simultaneously predicting frame-level detection boxes and the associated action class scores (§ 3.1-3.2);
- uses an original fusion strategy for merging appearance and motion cues based on the softmax probability scores and spatial overlaps of the detection bounding boxes (§ 3.3);
- brings forward a two-pass dynamic programming (DP) approach for constructing space time action tubes (§ 3.4).

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 (§ 4). Thanks to our two-pass action tube generation algorithm, in contrast to most existing action classification [24, 15, 23, 50, 51] and localisation [8, 54] approaches, our method is capable of detecting and localising multiple co-occurring action instances in temporally untrimmed videos (see Fig. 2).

## 2 Related work

Recently, inspired by the record-breaking performance of CNNs in image classification [1] and object detection from images [2], deep learning architectures have been increasingly applied to action classification [13, 15, 23], spatial [8] or spatio-temporal [54] action localisation, and event detection [57].

The action localisation problem, in particular, can be addressed by leveraging video segmentation methods. An example is the unsupervised greedy agglomerative clustering approach of [11], which resembles Selective Search space-time video blocks. Since [11] does not exploit the representative power of CNN features, they fail to achieve state-of-the-art results. Soomro *et al.* [26] learn the contextual relations between different space-time video segments. Such ‘supervoxels’, however, may end up spanning very long time intervals, failing to localise each action instance individually. Similarly, [29] uses unsupervised clustering to generate a small set of bounding box-like spatio-temporal action proposals. However, since the approach in [29] employs dense-trajectory features [50], it does not work on actions characterised by small motions [29].

The temporal detection of actions [9, 12] and gestures [3] in temporally untrimmed videos has also recently attracted much interest [9, 58]. Sliding window approaches have been extensively used [5, 19, 27, 52]. Unlike our approach, these methods [27, 52, 58] only address *temporal* detection, and suffer from the inefficient nature of temporal sliding windows. Our framework is based on incrementally linking frame-level region proposals and temporal smoothing (in a similar fashion to [9]), an approach which is computationally more efficient and can handle long untrimmed videos.

Indeed methods which connect frame-level region proposals for joint spatial and temporal localisation have risen to the forefront of current research. Gkioxari and Malik [8] have extended [2] and [23] to tackle action detection using unsupervised Selective-Search region proposals and separately trained SVMs. However, as the videos used to evaluate their work only contain one action and were already temporally trimmed (J-HMDB-21 [12]), it is not possible to assess their temporal localisation performance. Weinzaepfel *et al.*’s approach [54], instead, first generates region proposals using EdgeBoxes [40] at frame level to later use a tracking-by-detection approach based on a novel track-level descriptor called a Spatio-Temporal Motion Histogram. Moreover, [54] achieves temporal trimming using a multi-scale sliding window over each track, making it inefficient for longer video sequences. Our approach improves on both [8, 54] by using an efficient two-stage single network for detection of region proposals and two passes of dynamic programming for tube construction.

Some of the reviewed approaches [29, 54] could potentially be able to detect co-occurring actions. However, [54] limit their method to produce maximum of two detections per class, while [29] does so on the MSRII dataset [10] which only contains three action classes of repetitive nature (clapping, boxing and waving). Klaser *et al.* [16] use a space-time descriptor and a sliding window classifier to detect the location of only two actions (phoning and standing up). In contrast, in our LIRIS-HARL tests (§ 4) we consider 10 diverse action categories.

## 3 Methodology

As outlined in Figure 1, our approach combines a region-proposal network (§ 3.1-Fig. 1b) with a detection network (§ 3.2-Fig. 1d), and fuses the outputs (§ 3.3-Fig. 1f) to generate action tubes (§ 3.4-Fig. 1g-h). All components are described in detail below.



### 3.1 Region Proposal Network

To generate rectangular action region hypotheses in a video frame we adopt the Region Proposal Network (RPN) approach of [23], which is built on top of the last convolutional layer of the VGG-16 architecture by Simonyan and Zisserman [24]. 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’) [23] are generated. We consider those anchors with a high Intersection-over-Union ( $IoU$ ) 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.

### 3.2 Detection network

For the detection network we use a Fast R-CNN net [8] with a VGG-16 architecture [24]. This takes the RPN-based region proposals (§ 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 [23] to train both the RPN and Fast R-CNN networks. Shaoqing *et al.* [23] 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<sup>rd</sup> and 4<sup>th</sup> 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<sup>st</sup> and 2<sup>nd</sup> steps of [23], while neglecting the 3<sup>rd</sup> and 4<sup>th</sup> steps suggested by [23].

### 3.3 Fusion of appearance and motion cues

In a work by Redmon *et al.* [25], 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 (c.f. Fig. 1f).

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

$$s_c^*(\mathbf{b}_i^s) = s_c(\mathbf{b}_i^s) + s_c(\mathbf{b}_{max}^f) \times IoU(\mathbf{b}_i^s, \mathbf{b}_{max}^f). \quad (1)$$

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. In our tests we set  $\tau = 0.3$ .

### 3.4 Action tube generation

The output of our fusion stage (§ 3.3) is, for each video frame, a collection of detection boxes for each action category, together with their associated augmented classification scores (1). Detection boxes can then be linked up in time to identify video regions most likely to be associated with a single action instance, or *action tube*. Action tubes are connected sequences of detection boxes in time, without interruptions, and unlike those in [8] they are not constrained to span the entire video duration.

They are obtained as solutions to two consecutive energy maximisation problems. First a number of action-specific paths  $\mathbf{p}_c = \{\mathbf{b}_1, \dots, \mathbf{b}_T\}$ , spanning the entire video length, are constructed by linking detection boxes over time in virtue of their class-specific scores and their temporal overlap. Second, action paths are temporally trimmed by ensuring that the constituting boxes' detection scores are consistent with the foreground label  $c$ .

**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 pairwise potentials:

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

where  $s_c^*(\mathbf{b}_t)$  denotes the augmented score (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  $\lambda_o$  is a scalar parameter weighting the relative importance of the pairwise term. The value of the energy (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,  $\mathbf{p}_c^* = \operatorname{argmax}_{\mathbf{p}_c} E(\mathbf{p}_c)$ , by simply applying the Viterbi algorithm [8].

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

**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 (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  $\mathbf{L}_{\mathbf{p}_c} = [l_1, l_2, \dots, l_T]'$ : i) is consistent with the unary scores (1);

and ii) is smooth (no sudden jumps).

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

$$\mathbf{L}_{\mathbf{p}_c}^* = \underset{\mathbf{L}_{\mathbf{p}_c}}{\operatorname{argmax}} \left( \sum_{t=1}^T s_{l_t}(\mathbf{b}_t) - \lambda_l \sum_{t=2}^T \psi_l(l_t, l_{t-1}) \right), \quad (3)$$

where  $\lambda_l$  is a scalar parameter weighting the relative importance of the pairwise term. The pairwise potential  $\psi_l$  is defined to be:

$$\psi_l(l_t, l_{t-1}) = \begin{cases} 0 & \text{if } l_t = l_{t-1} \\ \alpha_c & \text{otherwise,} \end{cases} \quad (4)$$

where  $\alpha_c$  is a class-specific constant parameter which we set by cross validation. In the supplementary material, we show the impact of the class-specific  $\alpha_c$  on the detection accuracy. Equation (4) is the standard Potts model which penalises labellings that are not smooth, thus enforcing a piecewise constant solution. Again, we solve (3) 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 (1) of its constituting detection boxes.

## 4 Experimental validation and discussion

In order to evaluate our spatio-temporal action detection pipeline we selected what are currently considered among the most challenging action detection datasets: UCF-101 [25], LIRIS HARL D2 [66], and J-HMDB-21 [12]. UCF-101 is the largest, most diverse and challenging dataset to date, and contains realistic sequences with a large variation in camera motion, appearance, human pose, scale, viewpoint, clutter and illumination conditions. Although each video only contains a single action category, it may contain multiple action instances of the same action class. To achieve a broader comparison with the state-of-the-art, we also ran tests on the J-HMDB-21 [12] dataset. The latter is a subset of HMDB-51 [18] with 21 action categories and 928 videos, each containing a single action instance and trimmed to the action’s duration. The reported results were averaged over the three splits of J-HMDB-21. Finally we conducted experiments on the more challenging LIRIS-HARL dataset, which contains 10 action categories, including human-human interactions and human-object interactions (e.g., ‘discussion of two or several people’, and ‘a person types on a keyboard’<sup>2</sup>). In addition to containing multiple space-time actions, some of which occurring concurrently, the dataset contains scenes where relevant human actions take place amidst other irrelevant human motion.

For all datasets we used the exact same evaluation metrics and data splits as in the original papers. In the supplementary material, we further discuss all implementation details, and propose an interesting quantitative comparison between Selective Search- and RPN-generated region proposals.

**Performance comparison on UCF-101.** Table 1 presents the results we obtained on UCF-101, and compares them to the previous state-of-the-art [62, 69]. We achieve an mAP of

<sup>2</sup><http://liris.cnrs.fr/voir/activities-dataset>

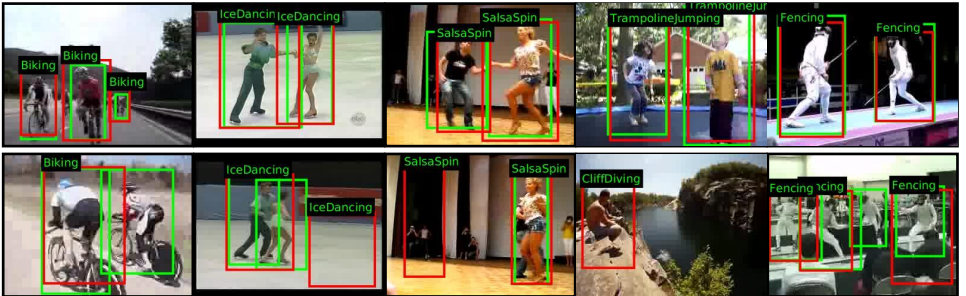
**Table 1: Quantitative action detection results (mAP) on the UCF-101 dataset.**

Spatio-temporal overlap threshold $\delta$	0.05	0.1	0.2	0.3	0.4	0.5	0.6
FAP [8]	42.80	—	—	—	—	—	—
STMH [54]	54.28	51.68	46.77	37.82	—	—	—
Our (appearance detection model)	68.74	66.13	56.91	48.28	39.10	30.67	22.77
Our (motion detection model)	67.04	64.86	57.33	47.45	38.65	28.90	19.49
<b>Our (appearance + motion fusion)</b>	<b>79.12</b>	<b>76.57</b>	<b>66.75</b>	<b>55.46</b>	<b>46.35</b>	<b>35.86</b>	<b>26.79</b>

66.75% compared to 46.77% reported by [54] (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 46.35%, (comparable to 46.77% [54] 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 1), their combination (§ 3.3) delivers a significant performance increase.

Some representative example results from UCF-101 are shown in Fig. 3. Our method can detect several (more than 2) action instances concurrently, as shown in Fig. 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’, ‘SalsaSpin’ and ‘IceDancing’, respectively, which all concern multiple inherently co-occurring action instances. Class-specific video APs on UCF-101 are reported in the supplementary material.

**Performance comparison on J-HMDB-21.** The results we obtained on J-HMDB-21 are presented in Table 2. Our method again outperforms the state-of-the-art, with an mAP increase of 18% and 11% at  $\delta = .5$  as compared to [8] and [54], 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 the supplementary material). Sample detections on J-HMDB-21 are shown in Figure 4. Also, we list our classification accuracy results on J-HMDB-21 in Table 3, where it can be seen that our method achieves an 8% gain compared to [8].



**Figure 3:** Action detection/localisation results on UCF101. Ground-truth boxes are in green, detection boxes in red. The top row shows correct detections, the bottom one contains examples of more mixed results. In the last frame, 3 out of 4 ‘Fencing’ instances are nevertheless correctly detected.

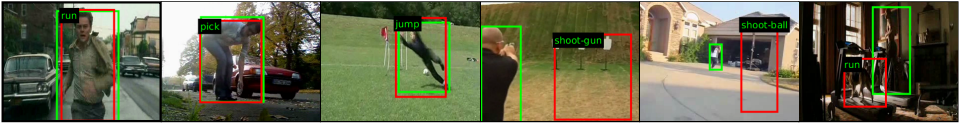


Figure 4: Sample space-time action localisation results on JHMDB. Left-most three frames: accurate detection examples. Right-most three frames: mis-detection examples.

Table 2: Quantitative action detection results (mAP) on the J-HMDB-21 dataset.

Spatio-temporal overlap threshold $\delta$	0.1	0.2	0.3	0.4	0.5	0.6	0.7
ActionTube [8] (mAP)	—	—	—	—	53.3	—	—
Wang et al. [65] (mAP)	—	—	—	—	56.4	—	—
STMH [66] (mAP)	—	63.1	63.5	62.2	60.7	—	—
Our (appearance detection model) (mAP)	52.99	52.94	52.57	52.22	51.34	49.55	45.65
Our (motion detection model) (mAP)	69.63	69.59	69.49	69.00	67.90	65.25	54.35
<b>Our (appearance+motion fusion) (mAP)</b>	<b>72.65</b>	<b>72.63</b>	<b>72.59</b>	<b>72.24</b>	<b>71.50</b>	<b>68.73</b>	<b>56.57</b>

Table 3: Classification accuracy on the J-HMDB-21 dataset.

Method	Wang et al. [65]	STMH [66]	ActionTube [8]	Our (appearance+motion fusion)
<b>Accuracy (%)</b>	56.6	61	62.5	<b>70.0</b>

**Performance comparison on LIRIS-HARL.** LIRIS HARL allows us to demonstrate the efficacy of our approach on temporally un-trimmed videos with co-occurring actions. For this purpose we use LIRIS-HARL’s specific evaluation tool - the results are shown in Table 4. Our results are compared with those of i) VPULABUAM-13 [22] and ii) IACAS-51 [10] 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 [65] and measuring temporal precision and recall along with spatial precision and recall, to produce an integrated score. We refer the readers to [65] for more details on LIRIS HARL’s evaluation metrics.



Figure 5: 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’. Left-most three frames: accurate detection examples. Right-most three frames: mis-detection examples.

We also report in Table 5 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 (§ 3.4), we also generate results (mAP) using only the first DP pass (§ 3.4). Without the 2<sup>nd</sup> pass performance decreases by 20%, highlighting the importance of temporal trimming in the construction of action tubes.

Table 4: Quantitative action detection results on the LIRIS-HARL dataset.

Method	Recall-10	Precision-10	F1-Score-10	$I_{sr}$	$I_{sp}$	$I_{tr}$	$I_{tp}$	IQ
VPULABUAM-13-IQ [22]	0.04	0.08	0.05	0.02	0.03	0.03	0.03	0.03
IACAS-51-IQ [10]	0.03	0.04	0.03	0.01	0.01	0.03	0.00	0.02
<b>(Ours)</b>	<b>0.568</b>	<b>0.595</b>	<b>0.581</b>	<b>0.5383</b>	<b>0.3402</b>	<b>0.4802</b>	<b>0.4739</b>	<b>0.458</b>

Table 5: Quantitative action detection results (mAP) on LIRIS-HARL for different  $\delta$ .

Spatio-temporal overlap threshold $\delta$	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
<b>Appearance+motion fusion</b> with two DP passes	<b>54.18</b>	<b>49.10</b>	<b>35.91</b>	<b>28.03</b>	<b>21.36</b>

**Test-time detection speed comparison.** Finally, we compared detection speed at test time of the combined region proposal generation and CNN feature extraction approach used in ([8], [34]) to our neural-net based, single stage action proposal and classification pipeline on the J-HMDB-21 dataset. We found our method to be  $10\times$  faster than [8] and  $5\times$  faster than [34], with a mean of 113.52 [8], 52.23 [34] and 10.89 (ours) seconds per video, averaged over all the videos in J-HMDB-21 split1. More timing comparison details and qualitative results (images and video clips) can be found in the supplementary material.

**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 ([28], [40]), our pipeline takes advantage of a supervised RPN-based region proposal approach which exhibits better recall values than [28] (supplementary-material). 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 (supplementary-material) where it is shown that the proposed fusion strategy (§ 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 and 6% on UCF-101 (supplementary-material). Additional qualitative results are provided in the supplementary video<sup>3</sup>, and on the project web page<sup>4</sup>, where the code has also been made available.

## 5 Conclusions and future work

In this paper, we presented a novel human action recognition approach which 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 it is capable of handling multiple concurrent action instances and temporally untrimmed videos.

Its combination of high accuracy and fast detection speed at test time is very promising for real-time 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 independent frames at test time and updating the dynamic programming optimisation step at every incoming frame.

**Acknowledgements.** This work was partly supported by ERC grant ERC-2012-AdG 321162-HELIOS, EPSRC grant Seebibyte EP/M013774/1 and EPSRC/MURI grant EP/N019474/1.

<sup>3</sup><https://www.youtube.com/embed/vBZsTgjhWaQ>

<sup>4</sup><http://sahasuman.bitbucket.org/bmvc2016>



## References

- [1] Liangliang Cao, Zicheng Liu, and Thomas S Huang. Cross-dataset action detection. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 1998–2005. IEEE, 2010.
- [2] Wei Chen, Caiming Xiong, Ran Xu, and Jason Corso. Actionness ranking with lattice conditional ordinal random fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 748–755, 2014.
- [3] Sergio Escalera, Xavier Baró, Jordi Gonzalez, Miguel A Bautista, Meysam Madadi, Miguel Reyes, Víctor Ponce-López, Hugo J Escalante, Jamie Shotton, and Isabelle Guyon. Chalearn looking at people challenge 2014: Dataset and results. In *Computer Vision-ECCV 2014 Workshops*, pages 459–473. Springer, 2014.
- [4] G. Evangelidis, G. Singh, and R. Horaud. Continuous gesture recognition from articulated poses. In *ECCV Workshops*, 2014.
- [5] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Temporal localization of actions with actoms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(11):2782–2795, 2013.
- [6] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.
- [7] Ross Girshick, Jeff Donahue, Trevor Darrel, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2014.
- [8] G Gkioxari and J Malik. Finding action tubes. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2015.
- [9] A Gorban, H Idrees, YG Jiang, A Roshan Zamir, I Laptev, M Shah, and R Sukthankar. Thumos challenge: Action recognition with a large number of classes, 2015.
- [10] Yonghao He, Hao Liu, Wei Sui, Shiming Xiang, and Chunhong Pan. Liris harl competition participant, 2012. Institute of Automation, Chinese Academy of Sciences, Beijing <http://liris.cnrs.fr/harl2012/results.html>.
- [11] Manan Jain, Jan Van Gemert, Hervé Jégou, Patrick Bouthemy, and Cees GM Snoek. Action localization with tubelets from motion. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 740–747. IEEE, 2014.
- [12] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. Black. Towards understanding action recognition. *Proc. Int. Conf. Computer Vision*, 2013.
- [13] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):221–231, Jan 2013.
- [14] YG 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://csrcv.ucf.edu/THUMOS14>, 2014.

- [15] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2014.
- [16] A. Kläser, M. Marszałek, C. Schmid, and A. Zisserman. Human focused action localization in video. In *International Workshop on Sign, Gesture, Activity*, 2010.
- [17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.
- [18] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2556–2563. IEEE, 2011.
- [19] Ivan Laptev and Patrick Pérez. Retrieving actions in movies. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015.
- [21] 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.
- [22] Juan C. SanMiguel and Sergio Suja. Liris harl competition participant, 2012. Video Processing and Understanding Lab, Universidad Autonoma of Madrid, Spain, <http://liris.cnrs.fr/harl2012/results.html>.
- [23] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems 27*, pages 568–576. Curran Associates, Inc., 2014.
- [24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [25] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human action classes from videos in the wild. Technical report, CRCV-TR-12-01, 2012.
- [26] Khurram Soomro, Haroon Idrees, and Mubarak Shah. Action localization in videos through context walk. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.
- [27] Yicong Tian, Rahul Sukthankar, and Mubarak Shah. Spatiotemporal deformable part models for action detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2642–2649. IEEE, 2013.
- [28] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *Int. Journal of Computer Vision*, 2013. URL <http://www.huppelen.nl/publications/selectiveSearchDraft.pdf>.

- [29] Jan C van Gemert, Mihir Jain, Ella Gati, and Cees GM Snoek. Apt: Action localization proposals from dense trajectories. In *BMVC*, volume 2, page 4, 2015.
- [30] H. Wang, A. Kläser, C. Schmid, and C. Liu. Action Recognition by Dense Trajectories. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2011.
- [31] Heng Wang and Cordelia Schmid. Action Recognition with Improved Trajectories. In *Proc. Int. Conf. Computer Vision*, pages 3551–3558, 2013.
- [32] Limin Wang, Yu Qiao, and Xiaoou Tang. Video action detection with relational dynamic-poselets. In *Computer Vision–ECCV 2014*, pages 565–580. Springer, 2014.
- [33] Limin Wang, Yu Qiao, Xiaoou Tang, and Luc Van Gool. Actionness estimation using hybrid fully convolutional networks. In *CVPR*, pages 2708–2717, 2016.
- [34] 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.
- [35] 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.
- [36] Christian Wolf, Julien Mille, Eric Lombardi, Oya Celiktutan, Mingyuan Jiu, Moez Baccouche, Emmanuel Dellandrea, Charles-Edmond Bichot, Christophe Garcia, and B. Sankur. The LIRIS Human activities dataset and the ICPR 2012 human activities recognition and localization competition. Technical Report RR-LIRIS-2012-004, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon, March 2012. URL <http://liris.cnrs.fr/publis/?id=5498>.
- [37] Zhongwen Xu, Yi Yang, and Alexander G Hauptmann. A discriminative cnn video representation for event detection. *arXiv preprint arXiv:1411.4006*, 2014.
- [38] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *arXiv preprint arXiv:1507.05738*, 2015.
- [39] Gang Yu and Junsong Yuan. Fast action proposals for human action detection and search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1302–1311, 2015.
- [40] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014*, pages 391–405. Springer, 2014.

# 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 [8, 9].

**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 [10]. The 3-channel optical flow values (i.e., flow- $x$ , flow- $y$  and the flow magnitude) were then used to construct ‘motion frames’ [9]. 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 [6] 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> [6], 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.



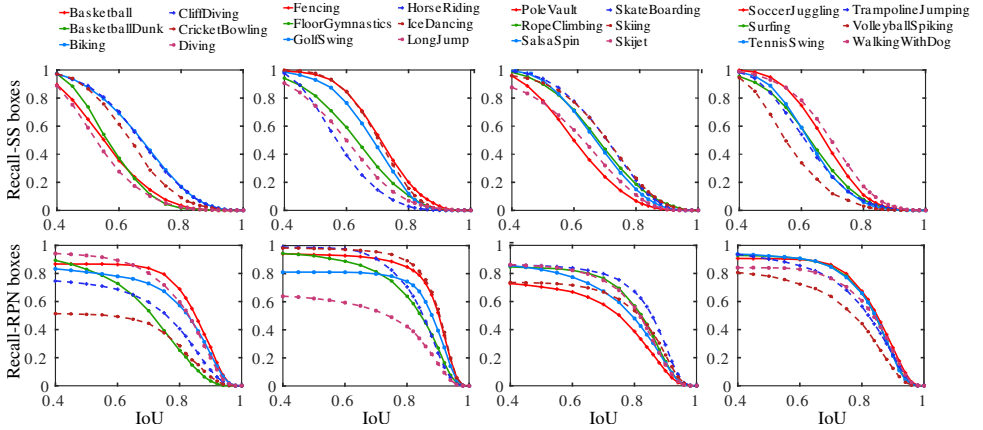


Figure 1: Performance comparison between Selective Search (SS) and RPN-based region proposals on four groups of action classes (vertical columns) in UCF-101. Top row: recall vs. IoU curve for SS. Bottom row: results for RPN-based region proposals.

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 [10, 11].

Training time: time computed on 2293 UCF-101 training video clips (split-1)			
ActionTube [10] and STMH [11]		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>
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 <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 [10] 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 [10, 11] 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 [10] and  $5\times$  than [11] in detecting actions in a video.

Table 5: Test time detection speed comparison on J-HMDB-21 with [10, 11].

ActionTube [10], STMH [11]	Average time (Sec./video)	Ours	Average time(Sec./video)
Selective Search [10] / EdgeBoxes [11]	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 [10] / 52.23 [11]</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.

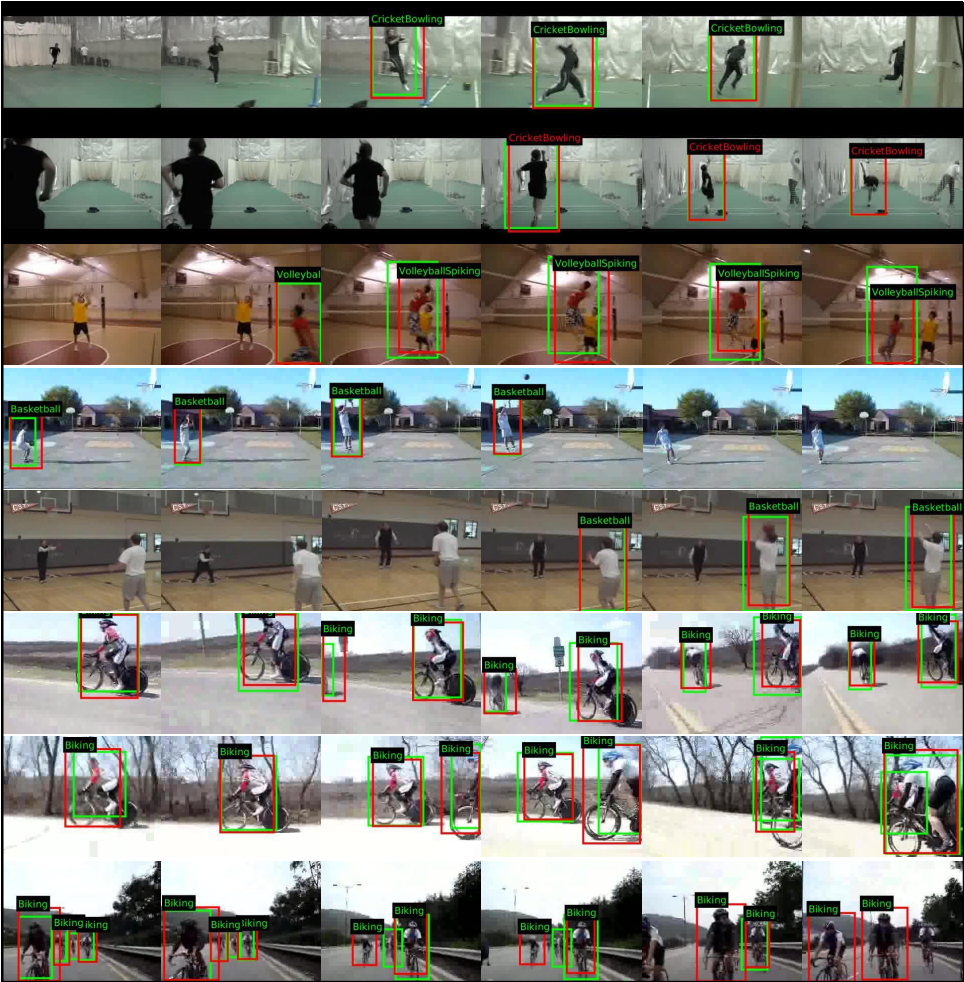


Figure 2: Sample spatio-temporal localisation results on UCF-101. Each row represents a UCF-101 test video clip. Ground-truth bounding boxes are in green, detection boxes in red.

## References

- [1] T Brox, A Bruhn, N Papenberg, 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.