Actions as Moving Points

Yixuan Li*, Zixu Wang*, Limin Wang**, and Gangshan Wu

State Key Laboratory for Novel Software Technology, Nanjing University, China

Abstract. The existing action tubelet detectors mainly depend on heuristic anchor design and placement, which might be computationally expensive and sub-optimal for precise localization of action instances. In this paper, we present a conceptually simple, computationally efficient, and more precise action tubelet detection framework, termed as *MovingCenter Detector* (MOC-detector), by treating an action instance as a trajectory of moving points. Based on the insight that movement information could simplify and assist the action tubelet detection, our MOC-detector is decomposed into three crucial head branches: (1) Center Branch for instance center detection and action recognition, (2) Movement Branch for movement estimation at adjacent frames to form trajectories of moving points, (3) Box Branch for spatial extent detection by directly regressing bounding box size at the estimated center point of each frame. These three branches work together to generate the tubelet detection results, which could be further linked to yield video-level tubes with a matching strategy. Our MOC-detector outperforms the existing state-of-the-art methods under the same setting for frame-mAP and video-mAP on the JHMDB and UCF101-24 datasets. The performance gap is more evident for higher video IoU, demonstrating that our MOC-detector is particularly useful for more precise action detection. We provide the code at https://github.com/MCG-NJU/MOC-Detector.

Keywords: Spatio-temporal action detection, anchor-free detection

1 Introduction

Spatio-temporal action detection is an important problem in video understanding, which aims to recognize all action instances present in a video and also localize them in both space and time. It has wide applications in many scenarios, such as video surveillance [20,12] and video captioning [31,36]. Some early approaches [7,21,25,32,33] apply an action detector at each frame independently and then link these frame-wise detection results across time using dynamic programming matching [7,26] or tracking [33]. These methods fail to well capture temporal information when conducting frame-level detection, and thus are less effective for detecting action tubes in reality. To address this issue, some approaches [24,14,11,35,38,27] try to perform action detection at the clip-level by

* Equal Contribution.
** Corresponding Author.

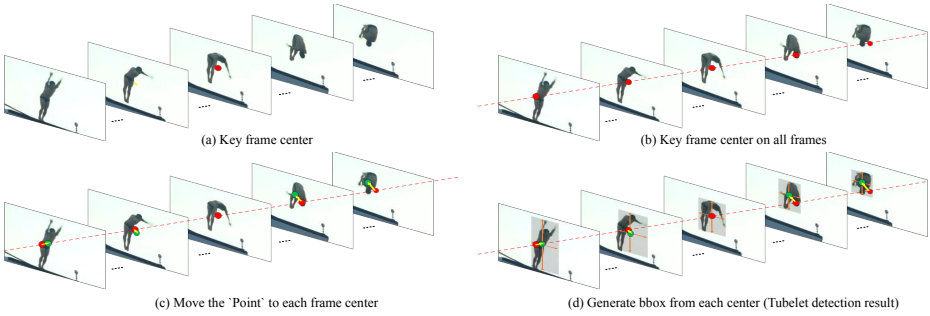

Fig. 1. Motivation Illustration. We focus on devising an action tubelet detector from a short sequence. Movement information naturally describes human behavior, and each action instance could be viewed as a trajectory of *moving points*. In this view, action tubelet detector could be decomposed into three simple steps: (1) localizing the center point (red dots) at key frame (i.e., center frame), (2) estimating the movement at each frame with respect to the center point (yellow arrows), (3) regressing bounding box size at the calculated center point (green dots) for all frames. Best viewed in color and zoom in.

exploiting short-term temporal information. In this sense, these methods input a sequence of frames and directly output detected tubelets (i.e., a short sequence of bounding boxes). This tubelet detection scheme yields a more principled and effective solution for video-based action detection and has shown promising results on standard benchmarks.

The existing tubelet detection methods [24,14,11,35,38,27] are closely related with the current mainstream object detectors such as Faster R-CNN [23] or SSD [19], which operate on a huge number of pre-defined anchor boxes. Although these anchor-based object detectors have achieved success in image domains, they still suffer from critical issues such as being sensitive to hyper-parameters (e.g., box size, aspect ratio, and box number) and less efficient due to densely placed bounding boxes. These issues are more serious when adapting the anchor-based detection framework from images to videos. First, the number of possible tubelet anchors would grow dramatically when increasing clip duration, which imposes a great challenge for both training and inference. Second, it is generally required to devise more sophisticated anchor box placement and adjustment to consider the variation along the temporal dimension. In addition, these anchor-based methods directly extend 2D anchors along the temporal dimension which predefine each action instance as a cuboid across space and time. This assumption lacks the flexibility to well capture temporal coherence and correlation of adjacent frame-level bounding boxes.

Inspired by the recent advances in anchor-free object detection [22,15,4,40,30], we present a **conceptually simple, computationally efficient, and more precise** action tubelet detector in videos, termed as *MovingCenter detector* (MOC-detector). As shown in Figure 1, our detector presents a new tubelet detection scheme by treating each instance as a trajectory of *moving points*. In

this sense, an action tubelet is represented by its center point in the key frame and offsets of other frames with respect to this center point. To determine the tubelet shape, we directly regress the bounding box size along the moving point trajectory on each frame. Our MOC-detector yields a fully convolutional one-stage tubelet detection scheme, which not only allows for more efficient training and inference but also could produce more precise detection results (as demonstrated in our experiments).

Specifically, our MOC detector decouples the task of tubelet detection into three sub-tasks: center detection, offset estimation and box regression. First, frames are fed into a 2D efficient backbone network for feature extraction. Then, we devise three separate branches: (1) Center Branch: detecting the action instance center and category; (2) Movement Branch: estimating the offsets of the current frame with respect to its center; (3) Box Branch: predicting bounding box size at the detected center point of each frame. This unique design enables three branches cooperate with each other to generate the tubelet detection results. Finally, we link these detected action tubelets across frames to yield long-range detection results following the common practice [14]. We perform experiments on two challenging action tube detection benchmarks of UCF101-24 [28] and JHMDB [13]. Our MOC-detector outperforms the existing state-of-the-art approaches for both frame-mAP and video-mAP on these two datasets, in particular for higher IoU criteria. Moreover, the fully convolutional nature of MOC detector yields a high detection efficiency of around 25FPS.

2 Related Work

2.1 Object Detection

Anchor-based Object Detectors. Traditional one-stage [19,22,17] and two-stage object detectors [6,9,5,23] heavily relied on predefined anchor boxes. Two-stage object detectors like Faster-RCNN [23] and Cascade-RCNN [1] devised RPN to generate RoIs from a set of anchors in the first stage and handled classification and regression of each RoI in the second stage. By contrast, typical one-stage detectors utilized class-aware anchors and jointly predicted the categories and relative spatial offsets of objects, such as SSD [19], YOLO [22] and RetinaNet [17].

Anchor-free Object Detectors. However, some recent works [30,40,15,4,41] have shown that the performance of anchor-free methods could be competitive with anchor-based detectors and such detectors also get rid of computation-intensive anchors and region-based CNN. CornerNet [15] detected object bounding box as a pair of corners, and grouped them to form the final detection. CenterNet [40] modeled an object as the center point of its bounding box and regressed its width and height to build the final result.

2.2 Spatio-temporal Action Detection

Frame-level Detector. Many efforts have been made to extend an image object detector to the task of action detection as frame-level action detec-

tors [7,32,21,25,26,33]. After getting the frame detection, linking algorithm is applied to generate final tubes. Although flows are used to capture motion information, frame-level detection fails to fully utilize the video's temporal information.

Clip-level Detector. In order to model temporal information for detection, some clip-level approaches or action tubelet detectors [14,11,35,16,38,27] have been proposed. ACT [14] took a short sequence of frames and output tubelets which were regressed from anchor cuboids. STEP [35] proposed a progressive method to refine the proposals over a few steps to solve the large displacement problem and utilized longer temporal information. Some methods [11,16] linked frame or tubelet proposals first to generate tubes proposal and then did classification.

These approaches are all based on anchor-based object detectors, whose design might be sensitive to anchor design and computationally cost due to large numbers of anchor boxes. Instead, we try to design an anchor-free action tubelet detector by treating each action instance as a trajectory of moving points. Experimental results demonstrate that our proposed action tubelet detector is effective for spatio-temporal action detection, in particular for the high video IoU.

3 Approach

Overview. Action tubelet detection aims at localizing a short sequence of bounding boxes from an input clip and recognizing its action category as well. We present a new tubelet detector, coined as MovingCenter detector (MOC-detector), by viewing an action instance as a trajectory of moving points. As shown in Figure 2, in our MOC-detector, we take a set of consecutive frames as input and separately feed them into an efficient 2D backbone to extract frame-level features. Then, we design three head branches to perform tubelet detection in an anchor-free manner. The first branch is Center Branch, which is defined on the center (key) frame. This center branch localizes the tubelet center and recognizes its action category. The second branch is Movement Branch, which is defined over all frames. This movement branch tries to relate adjacent frames to predict the center movement along the temporal dimension. The estimated movement would propagate the center point from key frame to other frames to generate a trajectory. The third branch is Box Branch, which operates on the detected center points of all frames. This branch focuses on determining the spatial extent of the detected action instance at each frame, by directly regressing the height and width of the bounding box. These three branches collaborate together to yield tubelet detection from a short clip, which will be further linked to form action tube detection in a long untrimmed video by following a common linking strategy [14]. We will first give a short description of the backbone design, and then provide technical details of three branches and the linking algorithm in the following subsections.

Backbone. In our MOC-detector, we input K frames and each frame is with the resolution of $W \times H$. First K frames are fed into a 2D backbone network

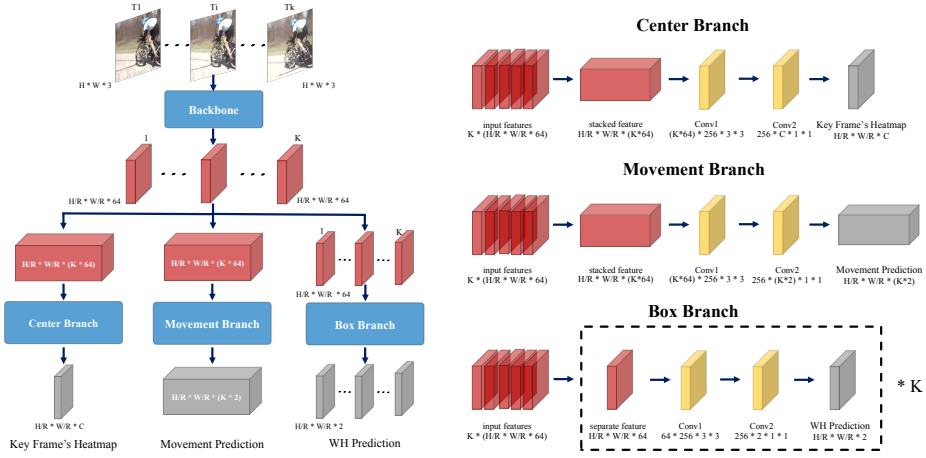

Fig. 2. Pipeline of MOC-detector. In the left, we present the overall MOC-detector framework. The red cuboids represent the extracted features, the blue boxes denote the backbone or detection head, and the gray cuboids are detection results produced by the center branch, the movement branch, the box branch. In the right, we show the detailed design of each branch. Each branch consists of a sequence of one 3*3 conv layer, one ReLu layer and one 1*1 conv layer, which is presented as yellow cuboids. The parameters of convolution are input channel, output channel, convolution kernel height, convolution kernel width.

sequentially to generate a feature volume $\mathbf{f} \in \mathbb{R}^{K \times \frac{W}{R} \times \frac{H}{R} \times B}$. R is the spatial downsample ratio and B denotes channel number. To keep the full temporal information for subsequent detection, we do not perform any downsampling over the temporal dimension. Specifically, we choose DLA-34 [37] architecture as our MOC-detector feature backbone following CenterNet [40]. This architecture employs an encoder-decoder architecture to extract features for each frame. The spatial downsampling ratio R is 4 and the channel number B is 64. The extracted features are shared by three head branches. Next we will present the technical details of these head branches.

3.1 Center Branch: Detect Center at Key Frame

The center branch aims at detecting the action instance center in the key frame (i.e., center frame) and recognizing its category based on the extracted video features. Temporal information is important for action recognition, and thereby we design a temporal module to estimate the action center and recognize its class by concatenating multi-frame feature maps along channel dimension. Specifically, based on the video feature representation $\mathbf{f} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times (K \times B)}$, we estimate a center heatmap $\hat{L} \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ for the key frame. The C is the number of action classes. The value of $\hat{L}_{x,y,c}$ represents the likelihood of detecting an action instance of class c at location (x, y), and higher value indicates a stronger

possibility. Specifically, we employ a standard convolution operation to estimate the center heatmap in a fully convolutional manner.

Training. We train the center branch following the common dense prediction setting [15,40]. For i^{th} action instance, we represent its center as key frame's bounding box center and utilize center's position for each action category as the ground truth label (x_{c_i}, y_{c_i}). We generate the ground truth heatmap $L \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ using a Gaussian kernel which produces the soft heatmap groundtruth $L_{x,y,c_i} = \exp(-\frac{(x-x_{c_i})^2+(y-y_{c_i})^2}{2\sigma_p^2})$. For other class (i.e., $c \neq c_i$), we set the heatmap $L_{x,y,c} = 0$. The σ_p is adaptive to instance size and we choose the maximum when two Gaussian of the same category overlap. We choose the training objective, which is a variant of focal loss [17], as follows:

$$\ell_{\text{center}} = -\frac{1}{n} \sum_{x,y,c} \begin{cases} (1-\hat{L}_{xyc})^{\alpha} \log(\hat{L}_{xyc}) & \text{if } L_{xyc} = 1 \\ (1-L_{xyc})^{\beta} (\hat{L}_{xyc})^{\alpha} \log(1-\hat{L}_{xyc}) & \text{otherwise} \end{cases} \tag{1}$$

where n is the number of ground truth instances and α and β are hyper-parameters of the focal loss [17]. We set $\alpha = 2$ and $\beta = 4$ following [15,40] in our experiments. It indicates that this focal loss is able to deal with the imbalanced training issue effectively [17].

Inference. After the training, the center branch could be deployed in tubelet detection for localizing action instance center and recognizing its category. Specifically, we detect all local peaks which are equal to or greater than their 8-connected neighbors in the estimated heatmap $\hat{L}$ for each class independently. And then keep the top N peaks from all categories as candidate centers with tubelet scores. Following [40], we set N as 100 and detailed ablation studies will be provided in Appendix A.

3.2 Movement Branch: Move Center Temporally

The movement branch tries to relate adjacent frames to predict the movement of the action instance center along the temporal dimension. Similar to center branch, movement branch also employs temporal information to regress the center offsets of current frame with respect to key frame. Specifically, movement branch takes stacked feature representation as input and outputs a movement prediction map $\hat{M} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times (K \times 2)}$. $2K$ channels represent center movements from key frame to current frames in X and Y directions. Given the key frame center $(\hat{x}_{key}, \hat{y}_{key})$, $\hat{M}_{\hat{x}_{key}, \hat{y}_{key}, 2j:2j+2}$ encodes center movement at j^{th} frame.

Training. The ground truth tubelet of i^{th} action instance is $[(x_{tl}^1, y_{tl}^1, x_{br}^1, y_{br}^1)$, $..., (x_{tl}^j, y_{tl}^j, x_{br}^j, y_{br}^j), ..., (x_{tl}^K, y_{tl}^K, x_{br}^K, y_{br}^K)]$, where subscript tl and br represent top-left and bottom-right points of bounding boxes, respectively. Let k be the key frame index, and the i^{th} action instance center at key frame is defined as follows:

$$(x_i^{key}, y_i^{key}) = (\lfloor (x_{tl}^k + x_{br}^k)/2 \rfloor, \lfloor (y_{tl}^k + y_{br}^k)/2 \rfloor). \tag{2}$$

We could compute the bounding box center (x_i^j, y_i^j) of i^{th} instance at j^{th} frame as follows:

$$(x_i^j, y_i^j) = ((x_{tl}^j + x_{br}^j)/2, (y_{tl}^j + y_{br}^j)/2). \tag{3}$$

Then, the ground truth movement of the i^{th} action instance is calculated as follows:

$$m_i = (x_i^1 - x_i^{key}, y_i^1 - y_i^{key}, ..., x_i^K - x_i^{key}, y_i^K - y_i^{key}). \tag{4}$$

For the training of movement branch, we optimize the movement map $\hat{M}$ only at the key frame center location and use the ℓ_1 loss as follows:

$$\ell_{\text{movement}} = \frac{1}{n}\sum_{i=1}^{n} |\hat{M}_{x_i^{key}, y_i^{key}} - m_i|. \tag{5}$$

Inference. After the movement branch training and given N detected action centers $\{(\hat{x}_i, \hat{y}_i)|i \in \{1, 2, \cdots, N\}\}$ from center branch, we obtain a set of movement vector $\{\hat{M}_{\hat{x}_i, \hat{y}_i}|i \in \{1, 2, \cdots, N\}\}$ for all detected action instance. Based on the results of movement branch and center branch, we could easily generate a trajectory set $T = \{T_i|i \in \{1, 2, \cdots, N\}\}$, and for the detected action center $(\hat{x}_i, \hat{y}_i)$, its trajectory of moving points is calculated as follows:

$$T_i = (\hat{x}_i, \hat{y}_i) + [\hat{M}_{\hat{x}_i, \hat{y}_i, 0:2}, \hat{M}_{\hat{x}_i, \hat{y}_i, 2:4}, \cdots, \hat{M}_{\hat{x}_i, \hat{y}_i, 2K-2:2K}]. \tag{6}$$

3.3 Box Branch: Determine Spatial Extent

The box branch is the last step of tubelet detection and focuses on determining the spatial extent of the action instance. Unlike center branch and movement branch, we assume box detection only depends on the current frame and temporal information will not benefit the class-agnostic bounding box generation. In this sense, this branch could be performed in a frame-wise manner. Specifically, box branch generates a size prediction map $\hat{S}^j \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$ for the j^{th} frame to directly estimate the bounding box size (i.e., width and height).

Training. The ground truth bbox size of i^{th} action instance at j^{th} frame can be represented as follows:

$$s_i^j = (x_{br}^j - x_{tl}^j, y_{br}^j - y_{tl}^j). \tag{7}$$

With this ground truth bounding box size, we optimize the Box Branch at the center points of all frames for each tubelet with ℓ_1 Loss as follows:

$$\ell_{\text{box}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{K} |\hat{S}_{p_i^j}^j - s_i^j|. \tag{8}$$

Note that the p_i^j is the i^{th} instance ground truth center at j^{th} frame. So the overall training objective of our MOC-detector is

$$\ell = \ell_{\text{center}} + a\ell_{\text{movement}} + b\ell_{\text{box}}, \tag{9}$$

where we set a=1 and b=0.1 in all our experiments. Detailed ablation studies will be provided in Appendix A.

Inference. We are ready to generate the tubelet detection results based on center trajectories T from movement branch and size prediction heatmap $\hat{S}$ for each location produced by this branch. For j^{th} point in trajectory T_i, we use (T_x, T_y) to denote its coordinates, and (w,h) to denote Box Branch size output $\hat{S}$ at specific location. Then, the bounding box for this point is calculated as:

$$(T_x - w/2, T_y - h/2, T_x + w/2, T_y + h/2). \tag{10}$$

3.4 Tubelet Linking

After introducing the technical details of MOC, we are ready to describe how to link the MOC-detector results to obtain tubes spanning different extents of the video. Note that MOC determines temporal extent in the linking stage with post-processing.

As our main goal is to propose a new tubelet detector and to fairly compare with previous methods, we use the same linking algorithm in [14]. Given a video, MOC extracts tubelets for each sequence of K frames and keeps top 10 tublets as candidates, which are linked into the final tubes in a frame by frame manner. **Initialization:** In the first frame, every candidate starts a new link. At a given frame, candidates which are not assigned to any existing links start new links. **Linking:** one candidate can only be assigned to one existing link when it meets three conditions: (1) the candidate is not selected by other links, (2) the candidate t has the highest score, (3) the overlap between link and candidate is greater than a threshold τ. **Termination:** An existing link stops if it has not been extended in consecutive K frames. Initialization and termination determine tubes' temporal extents. Tubes with low confidence and short duration are abandoned. More details are described in [14].

4 Experiments

4.1 Datasets and Implementation Details

To verify the effectiveness of MOC-detector for video-based action detection, we perform experiments on two challenging benchmarks: UCF101-24 [28] and JHMDB [13]. We notice that AVA [8] is a larger dataset for action detection but only contains a single-frame action instance annotation for each 3s clip, which concentrates on detecting actions on a single key frame. Thus, AVA is not suitable to verify the effectiveness of tubelet action detectors.

UCF101-24. The UCF101 dataset is a common benchmark for action recognition and contains spatio-temporal action instance annotations for 3207 videos from 24 sports classes. This video dataset is untrimmed and thus more challenging for action detection. Following the common setting [21,14], we report the action detection performance for the first split only.

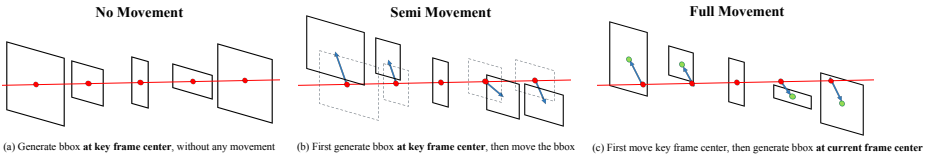

Fig. 3. Illustration of Three Movement Strategies. Note that the arrow represents moving according to Movement Branch prediction, the red dot represents the key frame center and the green dot represents the current frame center, which is localized by moving key frame center according to Movement Branch prediction.

JHMDB. The HMDB51 dataset is another smaller action recognition benchmark with 51 action classes. JHMDB is a subset of HMDB51, containing 928 videos from 21 action classes from our daily life. It is worth noting that these video clips are trimmed to the whole action instance. Thus the action detection on JHMDB mainly focuses on classification and spatial detection. We report results averaged over three splits following the common setting [21,14].

Evaluation Metrics. Following [33,7,14], we utilize frame mAP and video mAP to evaluate detection accuracy. Both frame-level and video-level AP calculations are based on IoU. The frame AP calculates the IoU based on the frame-level bounding box while the video AP calculates the IoU based on the video-level tube. The frame mAP is independent of the tube linking algorithm but the video mAP depends both on tubelet detection and linking algorithm. To better demonstrate the effectiveness of our MOC-detector on tubelet detection, we use the same linking algorithm with the previous method ACT [14].

Implementation Details. We choose the DLA34 [37] as our backbone with COCO [18] pretrain and ImageNet [3] pretrain. We provide MOC results with COCO pretrain without extra explanation. For a fair comparison, we provide two-stream results on two datasets with both COCO pretrain and ImageNet pretrain in Section 4.3. The frame is resized to 288×288. The spatial downsample ratio R is set to 4 and the resulted feature map size is 72×72. During training, we use the same data augmentation as [14] to the whole video: photometric transformation, scale jittering, and location jittering. We use Adam with a learning rate 5e-4 to optimize the overall objective. The learning rate adjusts to convergence on the validation set and it decreases by a factor of 10 when performance saturates. The iteration maximum is set to 12 epochs on UCF101-24 [28] and 20 epochs on JHMDB [13].

4.2 Ablation Studies

For efficient exploration, we perform experiments only using RGB input modality, COCO pretrain, and K as 5 without extra explanation. Without special specified, we use exactly the same training strategy in this subsection.

Effectiveness of Movement Branch. In MOC, Movement Branch impacts on both bbox's location and size. Movement Branch moves key frame center

Table 1. Exploration study on MOC detector design with various combinations of movement strategies on UCF101-24.

Method	Strategy		F-mAP@0.5 (%)	Video-mAP (%)			
	Move Center	Bbox Align		@0.2	@0.5	@0.75	0.5:0.95
No Movement			68.22	68.91	37.77	19.94	19.27
Semi Movement	✓		69.78	76.63	48.82	**27.05**	26.09
Full Movement (MOC)	✓	✓	**71.63**	**77.74**	**49.55**	27.04	**26.09**

to other frames to locate bbox center, named as Move Center strategy. Box Branch estimates bbox size on the current frame center, which is located by Movement Branch not the same with key frame, named as Bbox Align strategy. To explore the effectiveness of Movement Branch, we compare MOC with other two detector designs, called as *No Movement* and *Semi Movement*. We set the tubelet length $K = 5$ in all detection designs with the same training strategy. As shown in Figure 3, **No Movement** directly removes the Movement Branch and just generates the bounding box for each frame at the same location with key frame center. **Semi Movement** first generates the bounding box for each frame at the same location with key frame center, and then moves the generated box in each frame according to Movement Branch prediction. **Full Movement (MOC)** first moves the key frame center to the current frame center according to Movement Branch prediction, and then Box Branch generates the bounding box for each frame at its own center. The difference between Full Movement and Semi Movement is that they generate the bounding box at different locations: one at the real center, and the other at the fixed key frame center. The results are summarized in Table 1.

First, we observe that the performance gap between No Movement and Semi Movement is 1.56% for frame mAP@0.5 and 11.05% for video mAP@0.5. Frame mAP measures the detection quality without linking algorithm, while video mAP mainly evaluates the tube level error of long-term detection. We find that the Movement Branch has a relatively small influence on frame mAP, but contributes much to improve the video mAP. We ascribe this result to the fact that accumulating subtle errors of tubelets in the linking process will gradually deteriorate video level detection. So it demonstrates that the movement information is important for improving video mAP. *Second*, we can see that Full Movement performs slightly better than Semi Movement for both video mAP and frame mAP. Without Bbox Align, Box Branch estimates bbox size at key frame center for all frames, which causes a small performance drop with MOC. This small gap implies that Box Branch is relatively robust to the box center and estimating bbox size at small shifted location only brings a very slight performance difference.

Study on Movement Branch Design. In practice, in order to find an efficient way to capture center movements, we implement movement branch in several different ways. The first one is *Flow Guided Movement* strategy which utilizes optical flow between adjacent frames to move action instance center. The second strategy, *Cost Volume Movement*, is to directly compute the movement offset by

Table 2. Exploration study on the Movement Branch design on UCF101-24 [28]. Note that our MOC-detector adopts the Center Movement.

Method	F-mAP@0.5 (%)	Video-mAP (%)			
		@0.2	@0.5	@0.75	0.5:0.95
Flow Guided Movement	69.38	75.17	42.28	22.26	21.16
Cost Volume Movement	69.63	72.56	43.67	21.68	22.46
Accumulated Movement	69.40	75.03	46.19	24.67	23.80
Center Movement	**71.63**	**77.74**	**49.55**	**27.04**	**26.09**

constructing cost volume between key frame and current frame following [39], but this explicit computing fails to yield better results and is slower due to the constructing of cost volume. The third one is *Accumulated Movement* strategy which predicts center movement between consecutive frames instead of with respect to key frame. The fourth strategy, *Center Movement*, is to employ 3D convolutional operation to directly regress the offsets of the current frame with respect to key frame as illustrated in Section 3.2. The results are reported in Table 2.

We notice that the simple Center Movement performs best and choose it as Movement Branch design in our MOC-detector, which directly employs a 3D convolution to regress key frame center movement for all frames as a whole. We will analyze the fail reason for other three designs. For *Flow Guided Movement*, (i) Flow is not accurate and just represents pixel movement, while *Center Movement* is supervised by box movement. (ii) Accumulating adjacent flow to generate trajectory will enlarge error. For the *Cost Volume Movement*, (i) We explicitly calculate the correlation of the current frame with respect to key frame. When regressing the movement of the current frame, it only depends on the current correlation map. However, when directly regressing movement with 3D convolutions, the movement information of each frame will depend on all frames, which might contribute to more accurate estimation. (ii) As cost volume calculation and offset aggregation involve a correlation without extra parameters, it is observed that the convergence is much harder than *Center Movement*. For *Accumulated Movement*, this strategy also causes the issue of error accumulation and is more sensitive to the training and inference consistency. In this sense, the ground truth movement is calculated at the real bounding box center during training, while for inference, the current frame center is estimated from movement branch and might not be so precise, so that *Accumulated Movement* would bring large displacement to the ground truth.

Study on Input Sequence Duration. The temporal length K of the input clip is an important parameter in our MOC-detector. In this study, we report the RGB stream performance of MOC on UCF101-24 [28] by varying K from 1 to 9 and the experiment results are summarized in Table 3. We reduce the training batch size for K=7 and K=9 due to GPU memory limitation.

First, we notice that when $K = 1$, our MOC-detector reduces to the frame-level detector which obtains the worst performance, in particular for video mAP.

Table 3. Exploration study on the tubelet duration K on UCF101-24.

Tubelet Duration	F-mAP@0.5 (%)	Video-mAP (%)			
		@0.2	@0.5	@0.75	0.5:0.95
$K = 1$	68.33	65.47	31.50	15.12	15.54
$K = 3$	69.94	75.83	45.94	24.94	23.84
$K = 5$	71.63	77.74	49.55	27.04	26.09
$K = 7$	**73.14**	**78.81**	**51.02**	**27.05**	**26.51**
$K = 9$	72.17	77.94	50.16	26.26	26.07

Table 4. Comparison with the state of the art on JHMDB (trimmed) and UCF101-24 (untrimmed). Ours (MOC) with † is pretrained on ImageNet [3] and the other is pretrained on COCO [18]

Method	JHMDB					UCF101-24				
	Frame-mAP@0.5 (%)	Video-mAP (%)				Frame-mAP@0.5 (%)	Video-mAP (%)			
		@0.2	@0.5	@0.75	0.5:0.95		@0.2	@0.5	@0.75	0.5:0.95
2D Backbone										
Saha *et al.* 2016 [25]	-	72.6	71.5	43.3	40.0	-	66.7	35.9	7.9	14.4
Peng *et al.* 2016 [21]	58.5	74.3	73.1	-	-	39.9	42.3	-	-	-
Singh *et al.* 2017 [26]	-	73.8	72.0	44.5	41.6	-	73.5	46.3	15.0	20.4
Kalogeiton *et al.* 2017 [14]	65.7	74.2	73.7	52.1	44.8	69.5	76.5	49.2	19.7	23.4
Yang *et al.* 2019 [35]	-	-	-	-	-	75.0	76.6	-	-	-
Song *et al.* 2019 [27]	65.5	74.1	73.4	52.5	44.8	72.1	77.5	52.9	21.8	24.1
Zhao *et al.* 2019 [38]	-	-	74.7	53.3	45.0	-	78.5	50.3	22.2	24.5
Ours (MOC)†	68.0	76.2	75.4	68.5	54.0	76.9	81.3	**54.4**	29.5	**28.4**
Ours (MOC)	**70.8**	**77.3**	**77.2**	**71.7**	**59.1**	**78.0**	**82.8**	53.8	**29.6**	28.3
3D Backbone										
Hou *et al.* 2017 [11] (C3D)	61.3	**78.4**	76.9	-	-	41.4	47.1	-	-	-
Gu *et al.* 2018 [8] (I3D)	73.3	-	78.6	-	-	**76.3**	-	**59.9**	-	-
Sun *et al.* 2018 [29] (S3D-G)	**77.9**	-	**80.1**	-	-	-	-	-	-	-

This confirms the common assumption that frame-level action detector lacks consideration of temporal information for action recognition and thus it is worse than those tubelet detectors, which agrees with our basic motivation of designing an action tubelet detector. *Second*, we see that the detection performance will increase as we vary K from 1 to 7 and the performance gap becomes smaller when comparing $K = 5$ and $K = 7$. From $K = 7$ to $K = 9$, detection performance drops because predicting movement is harder for longer input length. According to the results, we set K=7 in our MOC.

4.3 Comparison with the State of the Art

Finally, we compare our MOC with the existing state-of-the-art methods on the trimmed JHMDB dataset and the untrimmed UCF101-24 dataset in Table 4. For a fair comparison, we also report two-stream results with ImageNet pretrain.

Our MOC gains similar performance on UCF101-24 for ImageNet pretrain and COCO pretrain, while COCO pretrain obviously improves MOC's performance on JHMDB because JHMDB is quite small and sensitive to the pretrain model. Our method significantly outperforms those frame-level action detectors [25,21,26] both for frame-mAP and video-mAP, which perform action detection at each frame independently without capturing temporal information. [14,35,38,27] are all tubelet detectors, our MOC outperforms them for all metrics on both datasets, and the improvement is more evident for high IoU video

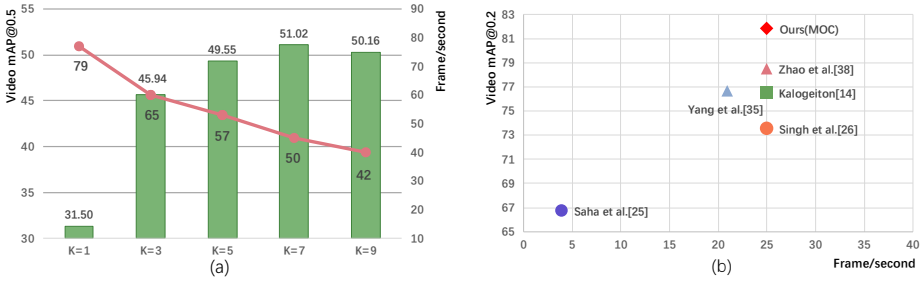

Fig. 4. Runtime Analysis and Comparison. (a) the detection results (green bars) and speeds (red dots) of MOC's RGB stream using different tubelet length K. (b) comparison with other methods. K=7 and two-stream fusion results.

mAP. This result confirms that our anchor-free MOC detector is more effective for localizing precise tubelets from clips than those anchor-based detectors, which might be ascribed to the flexibility and continuity of MOC detector by directly regressing tubelet shape. Our methods get comparable performance to those 3D backbone based methods [11,8,29]. These methods usually divide action detection into two steps: person detection (ResNet50-based Faster RCNN [23] pretrained on ImageNet), and action classification (I3D [2]/S3D-G [34] pretrained on Kinetics [2]+ROI pooling), and fail to provide a simple unified action detection framework. We provide a more detail comparison in Appendix C.

4.4 Runtime Analysis

MOC is an efficient detector and runs at 25 fps with tubelet length K=7 and two-stream fusion. Following ACT [14], we evaluate MOC's two-stream speed on a single NVIDIA TITAN Xp without flow computation, where temporal and spatial CNNs compute sequentially. Since all frames share the same backbone weights and Box Branch weights, we extract feature and estimate Box Branch just once for each frame, which avoids redundant computation for consecutive tubelets and makes MOC efficient. In Figure 4(a), detection accuracy improves but speed slows down with increasing K from 1 to 7. This trade-off between accuracy and efficiency can be adjusted according to specified preference. We roughly compare MOC's speed and accuracy with some existing methods which have reported speed in the original paper in Figure 4(b). [35,38,14] are all action tubelet detectors and our MOC gains more accurate detection results with comparable and even faster speed.

4.5 Visualization

In Figure 5, we give some qualitative examples to compare the performance between tubelet duration K=1 and K=7. Comparison between the second row

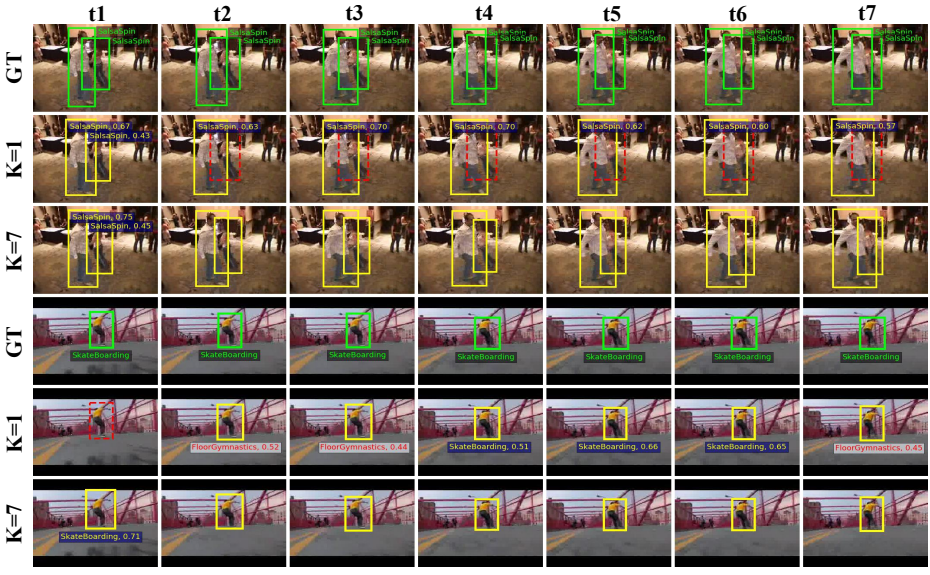

Fig. 5. Examples of Per-frame (K=1) and Tubelet (K=7) Detection. The yellow color boxes present detection results, whose categories and scores are provided beside. Yellow categories represent classifying correctly and red ones represent wrong. Red dashed boxes represent missed actors. Green boxes and categories are the ground truth. As our MOC-detector generates only one score and category for one tubelet, the score and category are only presented in the first frame for tubelet (K=7) detection. Note that we set the visualization threshold as 0.4.

and the third row shows that our tubelet detector leads to less missed detection results and localizes action more accurately owing to offset constraint in the same tubelet. What's more, comparison between the fifth and the sixth row presents that our tubelet detector can reduce classification error because some actions can not be discriminated by just looking one frame.

5 Conclusion and Future Work

In this paper, we have presented an action tubelet detector, termed as MOC-detector, by treating each action instance as a trajectory of moving points and directly regressing bounding box size at estimated center points of all frames. As demonstrated on two challenging datasets, the MOC-detector has brought a new state-of-the-art with both metrics of frame mAP and video mAP, while maintaining a reasonable computational cost. The superior performance is largely ascribed to the unique design of three branches and their cooperative modeling ability to perform tubelet detection. In the future, based on the proposed MOC-detector, we try to extend its framework to longer-term modeling and model action boundary in the temporal dimension, thus contributing to spatio-temporal action detection in longer continuous video streams.

Appendix A: Study on Hyper-parameters

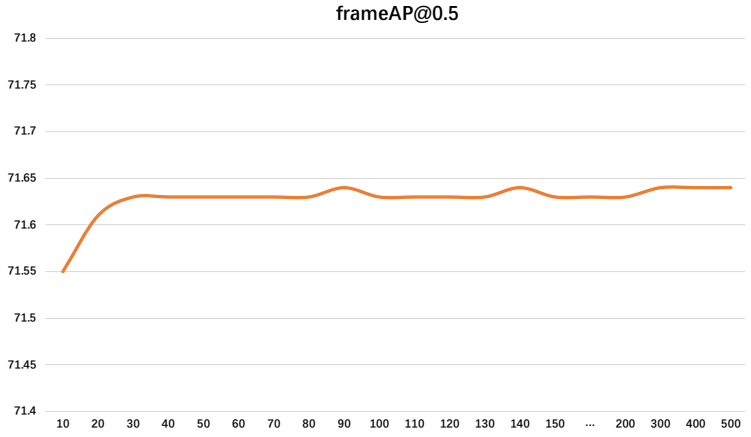

Fig. 6. Study on N. FrameAP@0.5 result on UCF101-24 [28] with tubelet length K=5 and only RGB input.

N in Center Branch. During inference, Center Branch keeps top N instances from all categories after max pooling operation, which is indicated in Section 3.1. We follow CenterNet [40], which is an anchor-free object detector and set N as 100. As shown in Figure 6, we can see that the detection result is robust to N and changes slightly after 40.

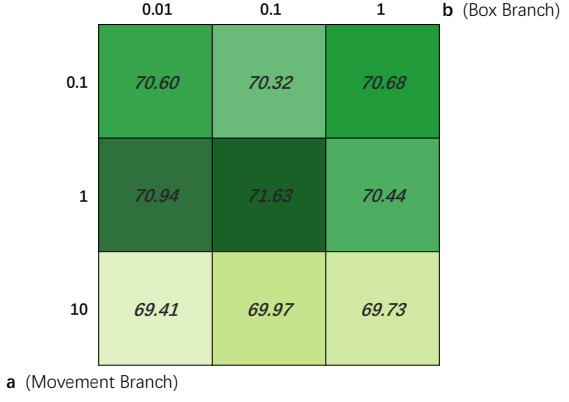

Fig. 7. Study on a and b. FrameAP@0.5 result on UCF101-24 [28] with tubelet length K=5 and only RGB input.

a and b in Loss Function. Equation 9 is MOCs training objective consisting of three branches loss. As shown in Figure 7, we have a linear search on a and b with tubelet length K=5 and only RGB input. We can see that a=1, b=0.1 performs best.

Appendix B: Error Analysis

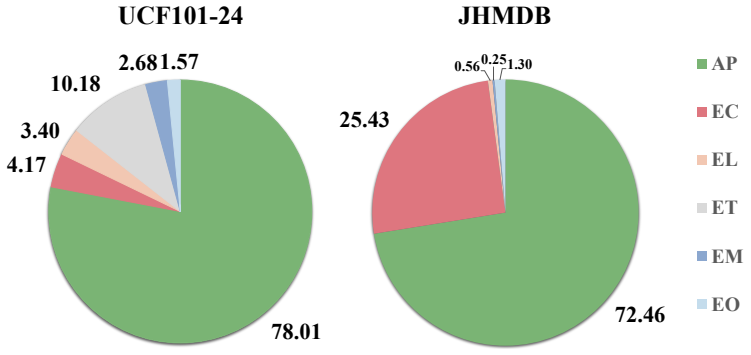

Fig. 8. Error analysis on UCF101-24 [28] and JHMDB [13] (only split 1). We report the detection error results according to five categories: (1) classification error E_C, (2) localization error E_L, (3) time error E_T, (4) missed detection E_M, and (5) other error E_O. The green part represents the correct detection. With tubelet length K=7 and two-stream fusion.

In this section, following [14], we conduct an error analysis on the frame mAP to better explore our proposed MOC-detector. In particular, we investigate five kinds of tubelet detection error: (1) classification error E_C: the detection IoU is greater than 0.5 with the ground-truth box of another action class. (2) localization error E_L: the detection class is correct in a frame but the bounding box IoU with ground truth is less than 0.5. (3) time error E_T: the detection in the untrimmed video covers the frame that doesn't belong to the temporal extent of the current action instance. (4) missed detection error E_M: cannot detect out a ground truth box. (5) other error E_O: the detection appears in a frame without the class and has IoU less than 0.5 with the ground truth bounding box of other classes.

We present error analysis on the untrimmed dataset UCF101-24 [28] and the trimmed dataset JHMDB [13] (only split 1) with tubelet length $K = 7$ and two-stream fusion. As shown in Figure 8, we find the major error is E_T, time error (10.18%), for the untrimmed dataset UCF101-24 [28] and E_C, classification error (25.43%), for the trimmed dataset JHMDB [13]. Although our MOC-detector has achieved state-of-art on both datasets, we will try to extend this framework to model longer temporal information to improve classification accuracy and model action boundary in the temporal dimension to eliminate time error.

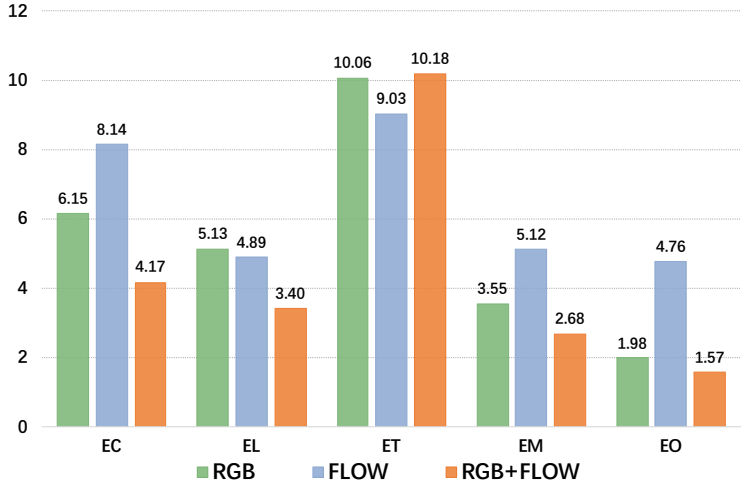

Fig. 9. Error Analysis with Two-stream Fusion. We report the detection error results according to five categories by changing input: (1) classification error E_C, (2) localization error E_L, (3) time error E_T, (4) missed detection E_M, and (5) other error E_O. With tubelet length K=7 and two-stream fusion on UCF101-24 [28].

We also visualize error analysis with two-stream fusion on UCF101-24 [28] and the results are reported in Figure 9. Note that we set tubelet length K as 7. First, spatial stream performs obviously better than the temporal stream for classification error and missed detection, owing to its richer information. Second, two-stream fusion improves the performance except for time error, which shows that two-stream fusion harms temporal localization.

Appendix C: More Results on JHMDB

Table 5. Comparison with Gu et al. [8] and Sun et al. [29] on JHMDB [13] (3 splits) with tubelet length K=7 and two stream fusion. Ours (MOC) † is pretrained on ImageNet [3] , Ours (MOC)†† is pretrained on COCO [18] and Ours (MOC)††† is pretrained on UCF101 [28] for action detection.

Method	GFLOPs	JHMDB				
		Frame-mAP@0.5 (%)	Video-mAP (%)			
			@0.2	@0.5	@0.75	0.5:0.95
Ours (MOC)†	29.4	68.0	76.2	75.4	68.5	54.0
Ours (MOC)††	29.4	70.8	77.3	77.2	71.7	59.1
Ours (MOC)†††	29.4	74.0	**80.7**	**80.5**	**75.0**	**60.2**
Gu *et al.* 2018 [8] (I3D)	>91.0	73.3	-	78.6	-	-
Sun *et al.* 2018 [29] (S3D-G)	>65.5	**77.9**	-	80.1	-	-

Our MOC is a one stage tubelet detector with 2D backbone. We compare it with two-stage detectors with 3D backbone [8,29] in Table 4, which perform comparably with us on UCF101 [28] while better than ours on JHMDB [13].

JHMDB [13] is really small and sensitive to the pretrain model. For fair comparison with 2D backbone methods in Table 4, we just provide results with ImageNet [3] pretrain and COCO [18] pretrain. But Gu et al [8] and Sun et al. [29] both pretrain 3D backbone on Kinetics [2], which is a large-scale video classification dataset and always boosts task results especially on small datasets. We pretrain our MOC on UCF101 [28] for action detection in Table 5, which outperforms Gu et al. [8] for all metrics with saving more than 3 times computation cost and performs comparably with Sun et al. [29] with saving more than 2 times computation cost. Note that Gu et al. [8] and Sun et al. [29] do not provide implementation code, so we just roughly estimate the backbone computation for each frame's detection result, whose input is 20 frames with resolution of 320*400. For Gu et al. [8], we calculate ResNet50 (conv4) [10] for action localization and I3D (Mixed_4e) [2] for classification. For Sun et al. [29] (Base Model), we calculate ResNet50 (conv4) [10] for action localization and S3D-G [34] for classification. For our MOC, we calculate the whole computation cost for each frame detection result. For fair comparison, we only use RGB as input for all methods.

References

1. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6154–6162 (2018)
2. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6299–6308 (2017)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
4. Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q.: Centernet: Keypoint triplets for object detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6569–6578 (2019)
5. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 1440–1448 (2015)
6. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014)
7. Gkioxari, G., Malik, J.: Finding action tubes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 759–768 (2015)
8. Gu, C., Sun, C., Ross, D.A., Vondrick, C., Pantofaru, C., Li, Y., Vijayanarasimhan, S., Toderici, G., Ricco, S., Sukthankar, R., et al.: Ava: A video dataset of spatio-temporally localized atomic visual actions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6047–6056 (2018)

9. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence **37**(9), 1904–1916 (2015)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
11. Hou, R., Chen, C., Shah, M.: Tube convolutional neural network (t-cnn) for action detection in videos. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5822–5831 (2017)
12. Hu, W., Tan, T., Wang, L., Maybank, S.: A survey on visual surveillance of object motion and behaviors. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **34**(3), 334–352 (2004)
13. Jhuang, H., Gall, J., Zuffi, S., Schmid, C., Black, M.J.: Towards understanding action recognition. In: Proceedings of the IEEE international conference on computer vision. pp. 3192–3199 (2013)
14. Kalogeiton, V., Weinzaepfel, P., Ferrari, V., Schmid, C.: Action tubelet detector for spatio-temporal action localization. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4405–4413 (2017)
15. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 734–750 (2018)
16. Li, D., Qiu, Z., Dai, Q., Yao, T., Mei, T.: Recurrent tubelet proposal and recognition networks for action detection. In: Proceedings of the European conference on computer vision (ECCV). pp. 303–318 (2018)
17. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
18. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
19. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016)
20. Oh, S., Hoogs, A., Perera, A., Cuntoor, N., Chen, C.C., Lee, J.T., Mukherjee, S., Aggarwal, J., Lee, H., Davis, L., et al.: A large-scale benchmark dataset for event recognition in surveillance video. In: CVPR 2011. pp. 3153–3160. IEEE (2011)
21. Peng, X., Schmid, C.: Multi-region two-stream r-cnn for action detection. In: European conference on computer vision. pp. 744–759. Springer (2016)
22. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
23. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
24. Saha, S., Singh, G., Cuzzolin, F.: Amtnet: Action-micro-tube regression by end-to-end trainable deep architecture. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4414–4423 (2017)
25. Saha, S., Singh, G., Sapienza, M., Torr, P.H., Cuzzolin, F.: Deep learning for detecting multiple space-time action tubes in videos. arXiv preprint arXiv:1608.01529 (2016)

26. Singh, G., Saha, S., Sapienza, M., Torr, P.H., Cuzzolin, F.: Online real-time multiple spatiotemporal action localisation and prediction. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3637–3646 (2017)
27. Song, L., Zhang, S., Yu, G., Sun, H.: Tacnet: Transition-aware context network for spatio-temporal action detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 11987–11995 (2019)
28. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)
29. Sun, C., Shrivastava, A., Vondrick, C., Murphy, K., Sukthankar, R., Schmid, C.: Actor-centric relation network. In: ECCV. pp. 335–351 (2018)
30. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
31. Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., Saenko, K.: Sequence to sequence-video to text. In: Proceedings of the IEEE international conference on computer vision. pp. 4534–4542 (2015)
32. Wang, L., Qiao, Y., Tang, X., Gool, L.V.: Actionness estimation using hybrid fully convolutional networks. In: CVPR. pp. 2708–2717 (2016)
33. Weinzaepfel, P., Harchaoui, Z., Schmid, C.: Learning to track for spatio-temporal action localization. In: Proceedings of the IEEE international conference on computer vision. pp. 3164–3172 (2015)
34. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 305–321 (2018)
35. Yang, X., Yang, X., Liu, M.Y., Xiao, F., Davis, L.S., Kautz, J.: Step: Spatio-temporal progressive learning for video action detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 264–272 (2019)
36. Yao, L., Torabi, A., Cho, K., Ballas, N., Pal, C., Larochelle, H., Courville, A.: Describing videos by exploiting temporal structure. In: Proceedings of the IEEE international conference on computer vision. pp. 4507–4515 (2015)
37. Yu, F., Wang, D., Shelhamer, E., Darrell, T.: Deep layer aggregation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2403–2412 (2018)
38. Zhao, J., Snoek, C.G.: Dance with flow: Two-in-one stream action detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9935–9944 (2019)
39. Zhao, Y., Xiong, Y., Lin, D.: Recognize actions by disentangling components of dynamics. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6566–6575 (2018)
40. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv preprint arXiv:1904.07850 (2019)
41. Zhou, X., Zhuo, J., Krahenbuhl, P.: Bottom-up object detection by grouping extreme and center points. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 850–859 (2019)