Single Shot Temporal Action Detection

Tianwei Lin[1], Xu Zhao[1,3,*], Zheng Shou[2]
[1]Department of Automation, Shanghai Jiao Tong University, China. [2]Columbia University, USA
[3]Cooperative Medianet Innovation Center (CMIC), Shanghai Jiao Tong University, China
{wzmsltw,zhaoxu}@sjtu.edu.cn,zs2262@columbia.edu

ABSTRACT

Temporal action detection is a very important yet challenging problem, since videos in real applications are usually long, untrimmed and contain multiple action instances. This problem requires not only recognizing action categories but also detecting start time and end time of each action instance. Many state-of-the-art methods adopt the "detection by classification" framework: first do proposal, and then classify proposals. The main drawback of this framework is that the boundaries of action instance proposals have been fixed during the classification step. To address this issue, we propose a novel Single Shot Action Detector (SSAD) network based on 1D temporal convolutional layers to skip the proposal generation step via directly detecting action instances in untrimmed video. On pursuit of designing a particular SSAD network that can work effectively for temporal action detection, we empirically search for the best network architecture of SSAD due to lacking existing models that can be directly adopted. Moreover, we investigate into input feature types and fusion strategies to further improve detection accuracy. We conduct extensive experiments on two challenging datasets: THUMOS 2014 and MEXaction2. When setting Intersection-over-Union threshold to 0.5 during evaluation, SSAD significantly outperforms other state-of-the-art systems by increasing mAP from 19.0% to 24.6% on THUMOS 2014 and from 7.4% to 11.0% on MEXaction2.

CCS CONCEPTS

• **Computing methodologies** → **Activity recognition and understanding**;

KEYWORDS

Temporal Action Detection, Untrimmed Video, SSAD network

1 INTRODUCTION

Due to the continuously booming of videos on the internet, video content analysis has attracted wide attention from both industry and academic field in recently years. An important branch of video content analysis is action recognition, which usually aims at classifying the categories of manually trimmed video clips. Substantial progress has been reported for this task in [6, 24, 36, 38, 40]. However, most videos in real world are untrimmed and may contain multiple action instances with irrelevant background scenes or activities. This problem motivates the academic community to put attention to another challenging task - temporal action detection. This task aims to detect action instances in untrimmed video, including temporal boundaries and categories of instances. Methods proposed for this task can be used in many areas such as surveillance video analysis and intelligent home care.

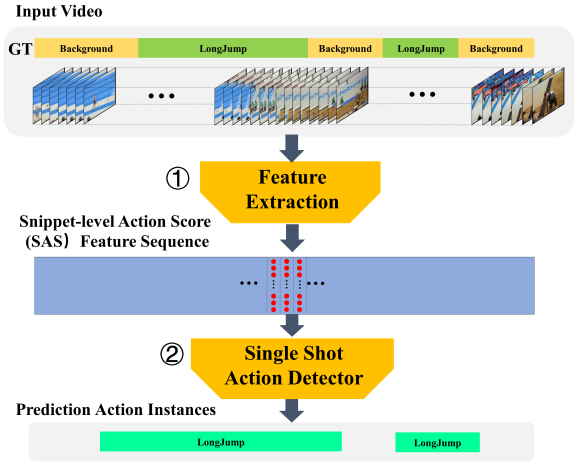

Figure 1: Overview of our system. Given an untrimmed long video, (1) we extract Snippet-level Action Score features sequence with multiple action classifiers; (2) SSAD network takes feature sequence as input and directly predicts multiple scales action instances without proposal generation step.

Temporal action detection can be regarded as a temporal version of object detection in image, since both of the tasks aim to determine the boundaries and categories of multiple instances (actions in time/ objects in space). A popular series of models in object detection are R-CNN and its variants [8, 9, 27], which adopt the "detect by classifying region proposals" framework. Inspired by R-CNN, recently many temporal action detection approaches adopt similar framework and classify temporal action instances generated by proposal method [3, 5, 29, 43] or simple sliding windows method [15, 23, 39]. This framework may has some major drawbacks: (1) proposal generation and classification procedures are separate and have to be trained separately, but ideally we want to train them in a joint manner to obtain an optimal model; (2) the proposal generation method or sliding windows method requires additional time consumption; (3) the temporal boundaries of action instances generated by the sliding windows method are usually approximative

This research has been supported by the funding from NSFC (61673269, 61273285) and the Cooperative Medianet Innovation Center (CMIC). * Corresponding author.

MM '17, Mountain View, CA, USA

DOI: 10.1145/3123266.3123343

rather than precise and left to be fixed during classification. Also, since the scales of sliding windows are pre-determined, it is not flexible to predict instances with various scales.

To address these issues, we propose the Single Shot Action Detector (SSAD) network, which is a temporal convolutional network conducted on feature sequence with multiple granularities. Inspired by another set of object detection methods - single shot detection models such as SSD [20] and YOLO [25, 26], our SSAD network skips the proposal generation step and directly predicts temporal boundaries and confidence scores for multiple action categories, as shown in Figure 1. SSAD network contains three sub-modules: (1) base layers read in feature sequence and shorten its temporal length; (2) anchor layers output temporal feature maps, which are associated with anchor action instances; (3) prediction layers generate categories probabilities, location offsets and overlap scores of these anchor action instances.

For better encoding of both spatial and temporal information in video, we adopt multiple action recognition models (action classifiers) to extract multiple granularities features. We concatenate the output categories probabilities from all action classifiers in snippet-level and form the Snippet-level Action Score (SAS) feature. The sequences of SAS features are used as input of SSAD network.

Note that it is non-trivial to adapt the single shot detection model from object detection to temporal action detection. Firstly, unlike VGGNet [31] being used in 2D ConvNet models, there is no existing widely used pre-trained temporal convolutional network. Thus in this work, we search multiple network architectures to find the best one. Secondly, we integrate key advantages in different single shot detection models to make our SSAD network work the best. On one hand, similar to YOLO9000 [26], we simultaneously predict location offsets, categories probabilities and overlap score of each anchor action instance. On the other hand, like SSD [20], we use anchor instances of multiple scale ratios from multiple scales feature maps, which allow network flexible to handle action instance with various scales. Finally, to further improve performance, we fuse the prediction categories probability with temporal pooled snippet-level action scores during prediction.

The main contributions of our work are summarized as follows:

(1) To the best of our knowledge, our work is the first Single Shot Action Detector (SSAD) for video, which can effectively predict both the boundaries and confidence score of multiple action categories in untrimmed video without the proposal generation step.

(2) In this work, we explore many configurations of SSAD network such as input features type, network architectures and post-processing strategy. Proper configurations are adopted to achieve better performance for temporal action detection task.

(3) We conduct extensive experiments on two challenging benchmark datasets: THUMOS'14 [14] and MEXaction2 [1]. When setting Intersection-over-Union threshold to 0.5 during evaluation, SSAD significantly outperforms other state-of-the-art systems by increasing mAP from 19.0% to 24.6% on THUMOS'14 and from 7.4% to 11.0% on MEXaction2.

2 RELATED WORK

Action recognition. Action recognition is an important research topic for video content analysis. Just as image classification network can be used in image object detection, action recognition models can be used in temporal action detection for feature extraction. We mainly review the following methods which can be used in temporal action detection. Improved Dense Trajectory (iDT) [37, 38] feature is consisted of MBH, HOF and HOG features extracted along dense trajectories. iDT method uses SIFT and optical flow to eliminate the influence of camera motion. Two-stream network [6, 30, 40] learns both spatial and temporal features by operating network on single frame and stacked optical flow field respectively using 2D Convolutional Neural Network (CNN) such as GoogleNet [35], VGGNet [31] and ResNet [12]. C3D network [36] uses 3D convolution to capture both spatial and temporal information directly from raw video frames volume, and is very efficient. Feature encoding methods such as Fisher Vector [38] and VAE [24] are widely used in action recognition task to improve performance. And there are many widely used action recognition benchmark such as UCF101 [34], HMDB51 [18] and Sports-1M [16].

Temporal action detection. This task focuses on learning how to detect action instances in untrimmed videos where the boundaries and categories of action instances have been annotated. Typical datasets such as THUMOS 2014 [14] and MEXaction2 [1] include large amount of untrimmed videos with multiple action categories and complex background information.

Recently, many approaches adopt "detection by classification" framework. For examples, many approaches [15, 23, 33, 39, 41] use extracted feature such as iDT feature to train SVM classifiers, and then classify the categories of segment proposals or sliding windows using SVM classifiers. And there are some approaches specially proposed for temporal action proposal [3, 5, 7, 22, 43]. Our SSAD network differs from these methods mainly in containing no proposal generation step.

Recurrent Neural Network (RNN) is widely used in many action detection approaches [21, 32, 42, 44] to encode feature sequence and make per-frame prediction of action categories. However, it is difficult for RNNs to keep a long time period memory in practice [32]. An alternative choice is temporal convolution. For example, Lea et al. [19] proposes Temporal Convolutional Networks (TCN) for temporal action segmentation. We also adopt temporal convolutional layers, which makes our SSAD network can handle action instances with a much longer time period.

Object detection. Deep learning approaches have shown salient performance in object detection. We will review two main set of object detection methods proposed in recent years. The representative methods in first set are R-CNN [9] and its variations [8, 27]. R-CNN uses selective search to generate multiple region proposals then apply CNN in these proposals separately to classify their categories; Fast R-CNN [8] uses a 2D RoI pooling layer which makes feature map be shared among proposals and reduces the time consumption. Faster RCNN [27] adopts a RPN network to generate region proposal instead of selective search.

Another set of object detection methods are single shot detection methods, which means detecting objects directly without generating proposals. There are two well known models. YOLO [25, 26] uses the whole topmost feature map to predict probabilities of multiple categories and corresponding confidence scores and location offsets. SSD [20] makes prediction from multiple feature map with

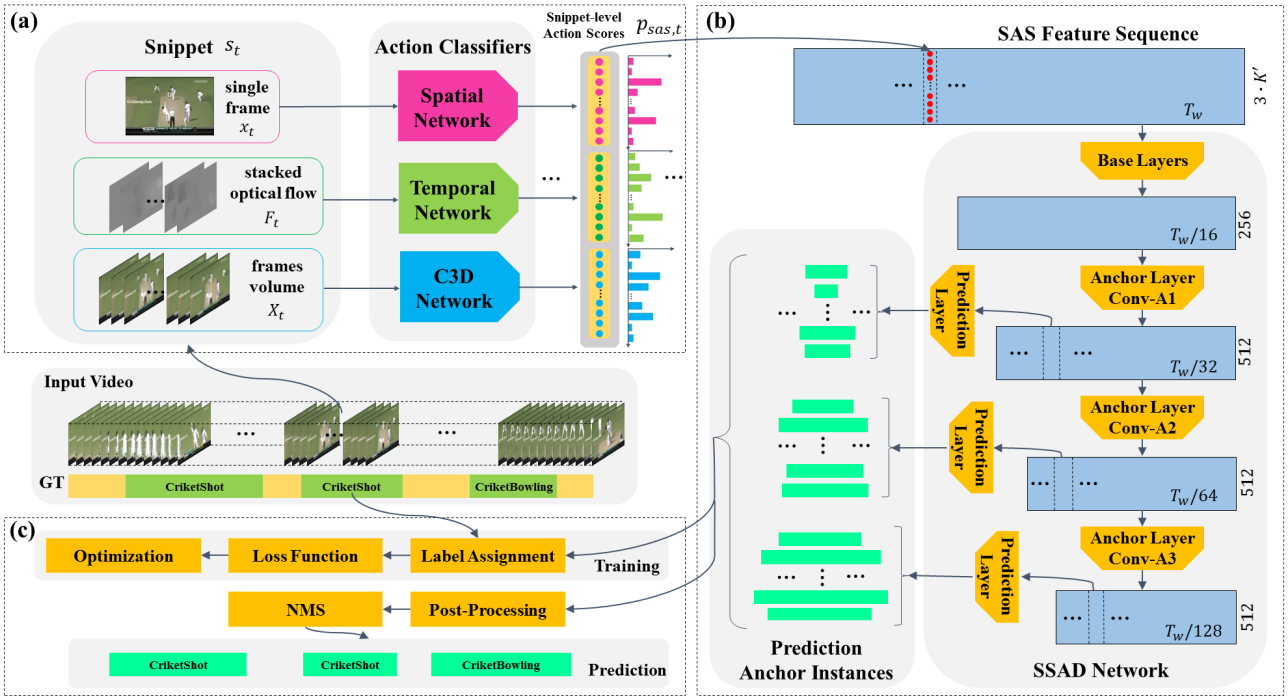

Figure 2: The framework of our approach. (a) Multiple action classifiers are used to extract Snippet-level Action Scores (SAS) feature. (b) The architecture of SSAD network: base layers are used to reduce the temporal dimension of input data; anchor layers output multiple scale feature map associated with anchor instances and prediction layers are used for predicting categories, location and confidence of anchor instances. (c) The training and prediction procedures: during training, we match anchor instances with ground truth instances and calculate loss function for optimization. During prediction, post-processing and NMS procedure are conducted on anchor instances to make final prediction.

multiple scales default boxes. In our work, we combine the characteristics of these single shot detection methods and embed them into the proposed SSAD network.

3 OUR APPROACH

In this section, we will introduce our approach in details. The framework of our approach is shown in Figure 2.

3.1 Problem Definition

We denote a video as $X_v = \{x_t\}_{t=1}^{T_v}$ where T_v is the number of frames in X_v and x_t is the t-th frame in X_v. Each untrimmed video X_v is annotated with a set of temporal action instances $\Phi_v = \{\phi_n = (\varphi_n, \varphi'_n, k_n)\}_{n=1}^{N_v}$, where N_v is the number of temporal action instances in X_v, and $\varphi_n, \varphi'_n, k_n$ are starting time, ending time and category of action instance ϕ_n respectively. $k_n \in \{1, ..., K\}$ where K is the number of action categories. Φ_v is given during training procedure and need to be predicted during prediction procedure.

3.2 Extracting of Snippet-level Action Scores

To apply SSAD model, first we need to make snippet-level action classification and get Snippet-level Action Score (SAS) features. Given a video X_v, a snippet $s_t = (x_t, F_t, X_t)$ is composed by three parts: x_t is the t-th frame in X_v, $F_t = \{f_{t'}\}_{t'=t-4}^{t+5}$ is stacked optical flow field derived around x_t and $X_t = \{x_{t'}\}_{t'=t-7}^{t+8}$ is video frames volume. So given a video X_v, we can get a sequence of snippets $S_v = \{s_t\}_{t=1}^{T_v}$. We pad the video X_v in head and tail with first and last frame separately to make S_v have the same length as X_v.

Action classifier. To evaluate categories probability of each snippet, we use multiple action classifiers with commendable performance in action recognition task: two-stream network [30] and C3D network [36]. Two-stream network includes spatial and temporal networks which operate on single video frame x_t and stacked optical flow field F_t respectively. We use the same two-stream network architecture as described in [40], which adopts VGGNet-16 network architecture. C3D network is proposed in [36], including multiple 3D convolution layers and 3D pooling layers. C3D network operates on short video frames volume X_t with length l, where l is the length of video clip and is set to 16 in C3D. So there are totally three individual action classifiers, in which spatial network measures the spatial information, temporal network measures temporal consistency and C3D network measures both. In section 4.3, we evaluate the effect of each action classifier and their combinations.

SAS feature. As shown in Figure 2(a), given a snippet s_t, each action classifier can generate a score vector $\boldsymbol{p_t}$ with length $K' = K + 1$, where K' includes K action categories and one background category. Then we concatenate output scores of each classifiers to form the **Snippet-level Action Score** (SAS) feature $\boldsymbol{p_{sas,t}} =$

$\left(\boldsymbol{p}_{S,t}, \boldsymbol{p}_{T,t}, \boldsymbol{p}_{C,t}\right)$, where $\boldsymbol{p}_{S,t}, \boldsymbol{p}_{T,t}, \boldsymbol{p}_{C,t}$ are output score of spatial, temporal and C3D network separately. So given a snippets sequence S_v with length T_v, we can extract a SAS feature sequence $P_v = \left\{\boldsymbol{p}_{sas,t}\right\}_{t=1}^{T_v}$. Since the number of frames in video is uncertain and may be very large, we use a large observation window with length T_w to truncate the feature sequence. We denote a window as $\omega = \{\varphi_\omega, \varphi'_\omega, P_\omega, \Phi_\omega\}$, where φ_ω and φ'_ω are starting and ending time of ω, P_ω and Φ_ω are SAS feature sequence and corresponding ground truth action instances separately.

3.3 SSAD Network

Temporal action detection is quite different from object detection in 2D image. In SSAD we adopt two main characteristics from single shot object detection models such as SSD [20] and YOLO [25, 26]: 1) unlike "detection by classification" approaches, SSAD directly predicts categories and location offsets of action instances in untrimmed video using convolutional prediction layers; 2) SSAD combine temporal feature maps from different convolution layers for prediction, making it possible to handle action instances with various length. We first introduce the network architecture.

Network architecture. The architecture of SSAD network is presented in Figure 2(b), which mainly contains three sub-modules: base layers, anchor layers and prediction layers. Base layers handle the input SAS feature sequence, and use both convolution and pooling layer to shorten the temporal length of feature map and increase the size of receptive fields. Then anchor layers use temporal convolution to continually shorten the feature map and output anchor feature map for action instances prediction. Each cell of anchor layers is associated with anchor instances of multiple scales. Finally, we use prediction layers to get classification score, overlap score and location offsets of each anchor instance.

In SSAD network, we adopt 1D temporal convolution and pooling to capture temporal information. We conduct Rectified Linear Units (ReLu) activation function [11] to output temporal feature map except for the convolutional prediction layers. And we adopt temporal max pooling since max pooling can enhance the invariance of small input change.

Base layers. Since there are no widely used pre-trained 1D ConvNet models such as the VGGNet [31] used in 2D ConvNet models, we search many different network architectures for SSAD network. These architectures only differ in base layers while we keep same architecture of anchor layers and prediction layers. As shown in Figure 3, we totally design 5 architectures of base layers. In these architectures, we mainly explore three aspects: 1) whether use convolution or pooling layer to shorten the temporal dimension and increase the size of receptive fields; 2) number of layers of network and 3) size of convolution layer's kernel. Notice that we set the number of convolutional filter in all base layers to 256. Evaluation results of these architectures are shown in section 4.3, and finally we adopt architecture B which achieves the best performance.

Multi-scale anchor layers. After processing SAS feature sequence using base layers, we stack three anchor convolutional layers (Conv-A1, Conv-A2 and Conv-A3) on them. These layers have same configuration: kernel size 3, stride size 2 and 512 convolutional filters. The output anchor feature maps of anchor layers are f_{A1}, f_{A2} and f_{A3} with size $(T_w/32 \times 512)$, $(T_w/64 \times 512)$ and

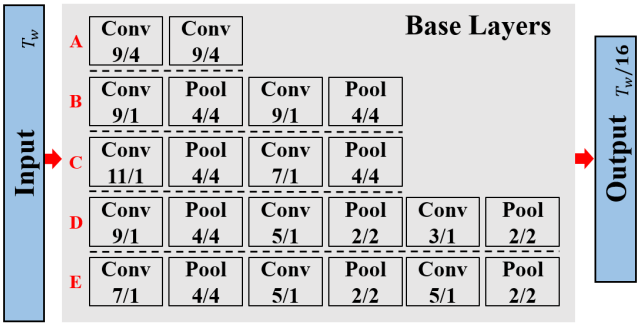

Figure 3: Multiple architectures of base layers. Input and output sizes are same for each architecture. Parameter of layer is shown with the format of *kernel*/*stride*. All convolutional layers have 512 convolutional filters. Evaluation results of these architectures are shown in section 4.3, and we adopt architecture B which achieves the best performance.

$(T_w/128 \times 512)$ separately. Multiple anchor layers decrease temporal dimension of feature map progressively and allow SSAD get predictions from multiple resolution feature map.

For each temporal feature map of anchor layers, we associate a set of multiple scale anchor action instances with each feature map cell as shown in Figure 4. For each anchor instance, we use convolutional prediction layers to predict overlap score, classification score and location offsets, which will be introduced later.

In term of the details of multi-scale anchor instances, the lower anchor feature map has higher resolution and smaller receptive field than the top anchor feature map. So we let the lower anchor layers detect short action instances and the top anchor layers detect long action instances. For a temporal feature map f of anchor layer with length M, we define base scale $s_f = \frac{1}{M}$ and a set of scale ratios $R_f = \{r_d\}_{d=1}^{D_f}$, where D_f is the number of scale ratios. We use $\{1, 1.5, 2\}$ for f_{A1} and $\{0.5, 0.75, 1, 1.5, 2\}$ for f_{A2} and f_{A3}. For each ratio r_d, we calculate $\mu_w = s_f \cdot r_d$ as anchor instance's default width. And all anchor instances associated with the m-th feature map cell share the same default center location $\mu_c = \frac{m+0.5}{M}$. So for an anchor feature map f with length M_f and D_f scale ratios, the number of associated anchor instances is $M_f \cdot D_f$.

Prediction layers. We use a set of convolutional filters to predict classification scores, overlap scores and location offsets of anchor instances associated with each feature map cell. As shown in Figure 4, for an anchor feature map f with length M_f and D_f scale ratios, we use $D_f \cdot (K' + 3)$ temporal convolutional filters with kernel size 3, stride size 1 for prediction. The output of prediction layer has size $\left(M_f \times \left(D_f \cdot (K' + 3)\right)\right)$ and can be reshaped into $\left(\left(M_f \cdot D_f\right) \times (K' + 3)\right)$. Each anchor instance gets a prediction score vector $\boldsymbol{p}_{\boldsymbol{pred}} = (\boldsymbol{p}_{\boldsymbol{class}}, p_{over}, \Delta c, \Delta w)$ with length $(K' + 3)$, where $\boldsymbol{p}_{\boldsymbol{class}}$ is classification score vector with length K', p_{over} is overlap score and Δc, Δw are location offsets. Classification score p_{class} is used to predict anchor instance's category. Overlap score p_{over} is used to estimate the overlap between anchor instance and ground truth instances and should have value between $[0, 1]$, so it is normalized by using sigmoid function:

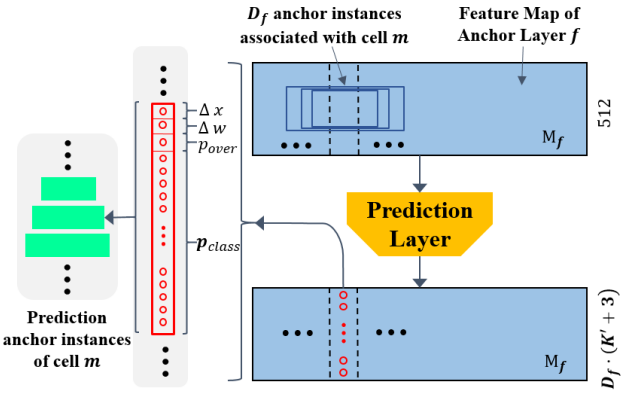

Figure 4: Anchor instances and prediction layer in temporal feature map. In feature map of a anchor layer, we associate a set of multiple scale anchor instances with each feature map cell. We use convolutional prediction layer to predict location offset, confidence and classification scores simultaneously for each anchor instance.

$$p'_{over} = sigmoid(p_{over}). \tag{1}$$

And location offsets Δc, Δw are used for adjusting the default location of anchor instance. The adjusted location is defined as:

$$\begin{aligned} \varphi_c &= \mu_c + \alpha_1 \cdot \mu_w \cdot \Delta c \\ \varphi_w &= \mu_w \cdot exp(\alpha_2 \cdot \Delta w), \end{aligned} \tag{2}$$

where φ_c and φ_w are center location and width of anchor instance respectively. α_1 and α_2 are used for controlling the effect of location offsets to make prediction stable. We set both α_1 and α_2 to 0.1. The starting and ending time of action instance are $\varphi = \varphi_c - \frac{1}{2} \cdot \varphi_w$ and $\varphi' = \varphi_c + \frac{1}{2} \cdot \varphi_w$ respectively. So for a anchor feature map f, we can get a anchor instances set $\Phi_f = \{\phi_n = (\varphi_c, \varphi_w, p_{class}, p'_{over})\}_{n=1}^{N_f}$, where $N_f = M_f \cdot D_f$ is the number of anchor instances. And the total prediction instances set is $\Phi_p = \left\{\Phi_{f_{A1}}, \Phi_{f_{A2}}, \Phi_{f_{A3}}\right\}$.

3.4 Training of SSAD network

Training data construction. As described in Section 3.2, for an untrimmed video X_v with length T_v, we get SAS features sequence P_v with same length. Then we slide window of length T_w in feature sequence with 75% overlap. The overlap of sliding window is aim to handle the situation where action instances locate in boundary of window and also used to increase the amount of training data. During training, we only keep windows containing at least one ground-truth instance. So given a set of untrimmed training videos, we get a training set $\Omega = \{\omega_n\}_{n=1}^{N_\omega}$, where N_ω is the number of windows. We randomly shuffle the data order in training set to make the network converge faster, where same random seed is used during evaluation.

Label assignment. During training, given a window ω, we can get prediction instances set Φ_p via SSAD network. We need to match them with ground truth set Φ_ω for label assignment. For an anchor instance ϕ_n in Φ_p, we calculate it's IoU overlap with all ground truth instances in Φ_ω. If the highest IoU overlap is higher than 0.5, we match ϕ_n with corresponding ground truth instance ϕ_g and regard it as positive, otherwise negative. We expand ϕ_n with matching information as $\phi'_n = \left(\varphi_c, \varphi_w, \boldsymbol{p_{class}}, p'_{over}, k_g, g_{iou}, g_c, g_w\right)$, where k_g is the category of ϕ_g and is set to 0 for negative instance, g_{iou} is the IoU overlap between ϕ_n and ϕ_g, g_c and g_w are center location and width of ϕ_g respectively. So a ground truth instance can match multiple anchor instances while a anchor instance can only match one ground truth instance at most.

Hard negative mining. During label assignment, only a small part of anchor instances match the ground truth instances, causing an imbalanced data ratio between the positive and negative instances. Thus we adopt the hard negative mining strategy to reduce the number of negative instances. Here, the hard negative instances are defined as negative instances with larger overlap score than 0.5. We take all hard negative instances and randomly sampled negative instances in remaining part to make the ratio between positive and negative instances be nearly 1:1. This ratio is chosen by empirical validation. So after label assignment and hard negative mining, we get $\Phi'_p = \{\phi'_n\}_{n=1}^{N_{train}}$ as the input set during training, where N_{train} is the number of total training instances and is the sum of the number of positives N_{pos} and negatives N_{neg}.

Objective for training. The training objective of the SSAD network is to solve a multi-task optimization problem. The overall loss function is a weighted sum of the classification loss (class), the overlap loss (conf), the detection loss (loc) and L2 loss for regularization:

$$L = L_{class} + \alpha \cdot L_{over} + \beta \cdot L_{loc} + \lambda \cdot L_2(\Theta), \tag{3}$$

where α, β and λ are the weight terms used for balancing each part of loss function. Both α and β are set to 10 and λ is set to 0.0001 by empirical validation. For the classification loss, we use conventional softmax loss over multiple categories, which is effective for training classification model and can be defined as:

$$L_{class} = L_{softmax} = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} (-log(P_i^{(k_g)})), \tag{4}$$

where $P_i^{(k_g)} = \frac{exp(p_{class,i}^{(k_g)})}{\sum_j exp(p_{class,i}^{(k_j)})}$ and k_g is the label of this instance.

L_{over} is used to make a precise prediction of anchor instances' overlap IoU score, which helps the procedure of NMS. The overlap loss adopts the mean square error (MSE) loss and be defined as:

$$L_{over} = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} (p'_{over,i} - g_{iou,i}). \tag{5}$$

L_{loc} is the Smooth L1 loss [8] for location offsets. We regress the center (ϕ_c) and width (ϕ_w) of predicted instance:

$$L_{loc} = \frac{1}{N_{pos}} \sum_{i=1}^{N_{pos}} (SL_1(\phi_{c,i} - g_{c,i}) + SL_1(\phi_{w,i} - g_{w,i})), \tag{6}$$

where $g_{c,i}$ and $g_{w,i}$ is the center location and width of ground truth instance. $L_2(\Theta)$ is the L2 regularization loss where Θ stands for the parameter of the whole SSAD network.

3.5 Prediction and post-processing

During prediction, we follow the aforementioned data preparation method during the training procedure to prepare test data, with the following two changes: (1) the overlap ratio of window is reduced to 25% to increase the prediction speed and reduce the redundant predictions; (2) instead of removing windows without annotation, we keep all windows during prediction because the removing operation is actually a leak of annotation information. If the length of input video is shorter than T_w, we will pad SAS feature sequence to T_w so that there is at least one window for prediction. Given a video X_v, we can get a set of $\Omega = \{\omega_n\}_{n=1}^{N_\omega}$. Then we use SSAD network to get prediction anchors of each window and merge these prediction as $\Phi_p = \{\phi_n\}_{n=1}^{N_p}$, where N_p is the number of prediction instances. For a prediction anchor instance ϕ_n in Φ_p, we calculate the mean Snippet-level Action Score $\bar{\boldsymbol{p}}_{sas}$ among the temporal range of instance and multiple action classifiers.

$$\bar{\boldsymbol{p}}_{sas} = \frac{1}{3 \cdot (\varphi' - \varphi)} \sum_{t=\varphi}^{\varphi'} \left(\boldsymbol{p}_{S,t} + \boldsymbol{p}_{T,t} + \boldsymbol{p}_{C,t} \right), \tag{7}$$

where φ and φ' are starting and ending time of prediction anchor instance respectively. Then we fuse categories scores $\bar{\boldsymbol{p}}_{sas}$ and $\boldsymbol{p}_{class}$ with multiplication factor p_{conf} and get the $\boldsymbol{p}_{final}$:

$$\boldsymbol{p}_{final} = p'_{over} \cdot (\boldsymbol{p}_{class} + \bar{\boldsymbol{p}}_{sas}). \tag{8}$$

We choose the maximum dimension k_p in $\boldsymbol{p}_{final}$ as the category of ϕ_n and corresponding score p_{conf} as the confidence score. We expand ϕ_n as $\phi'_n = \left\{\varphi_c, \varphi_w, p_{conf}, k_p\right\}$ and get prediction set $\Phi'_p = \{\phi'_n\}_{n=1}^{N_p}$. Then we conduct non-maximum suppress (NMS) in these prediction results to remove redundant predictions with confidence score p_{conf} and get the final prediction instances set $\Phi''_p = \{\phi'_n\}_{n=1}^{N_{p'}}$, where $N_{p'}$ is the number of the final prediction anchors. Since there are little overlap between action instances of same category in temporal action detection task, we take a strict threshold in NMS, which is set to 0.1 by empirical validation.

4 EXPERIMENTS

4.1 Dataset and setup

THUMOS 2014 [14]. The temporal action detection task of THUMOS 2014 dataset is challenging and widely used. The training set is the UCF-101 [34] dataset including 13320 trimmed videos of 101 categories. The validation and test set contain 1010 and 1574 untrimmed videos separately. In temporal action detection task, only 20 action categories are involved and annotated temporally. We only use 200 validation set videos (including 3007 action instances) and 213 test set videos (including 3358 action instances) with temporal annotation to train and evaluate SSAD network.

MEXaction2 [1]. There are two action categories in MEXaction2 dataset: "HorseRiding" and "BullChargeCape". This dataset is consisted of three subsets: YouTube clips, UCF101 Horse Riding clips and INA videos. YouTube and UCF101 Horse Riding clips are trimmed and used for training set, whereas INA videos are untrimmed with approximately 77 hours in total and are divided into training, validation and testing set. Regarding to temporal annotated action instances, there are 1336 instances in training set, 310 instances in validation set and 329 instances in testing set.

Evaluation metrics. For both datasets, we follow the conventional metrics used in THUMOS'14, which evaluate Average Precision (AP) for each action categories and calculate mean Average Precision (mAP) for evaluation. A prediction instance is correct if it gets same category as ground truth instance and its temporal IoU with this ground truth instance is larger than IoU threshold θ. Various IoU thresholds are used during evaluation. Furthermore, redundant detections for the same ground truth are forbidden.

4.2 Implementation Details

Action classifiers. To extract SAS features, action classifiers should be trained first, including two-stream networks [40] and C3D network [36]. We implement both networks based on Caffe [13]. For both MEXaction and THUMOS'14 datasets, we use trimmed videos in training set to train action classifier.

For spatial and temporal network, we follow the same training strategy described in [40] which uses the VGGNet-16 pre-trained on ImageNet [4] to intialize the network and fine-tunes it on training set. And we follow [36] to train the C3D network, which is pre-trained on Sports-1M [16] and then is fine-turned on training set.

SSAD optimization. For training of the SSAD network, we use the adaptive moment estimation (Adam) algorithm [17] with the aforementioned multi-task loss function. Our implementation is based on Tensorflow [2]. We adopt the Xavier method [10] to randomly initialize parameters of whole SSAD network because there are no suitable pre-trained temporal convolutional network. Even so, the SSAD network can be easily trained with quick convergence since it has a small amount of parameters (20 MB totally) and the input of SSAD network - SAS features are concise high-level feature. The training procedure takes nearly 1 hour on THUMOS'14 dataset.

Table 1: mAP results on THUMOS'14 with various IoU threshold θ used in evaluation.

θ	0.5	0.4	0.3	0.2	0.1
Karaman et al. [15]	0.2	0.3	0.5	0.9	1.5
Wang et al. [39]	8.5	12.1	14.6	17.8	19.2
Oneata et al. [23]	15.0	21.8	28.8	36.2	39.8
Richard et al. [28]	15.2	23.2	30.0	35.7	39.7
Yeung et al. [42]	17.1	26.4	36.0	44.0	48.9
Yuan et al. [44]	18.8	26.1	33.6	42.6	**51.4**
Shou et al. [29]	19.0	28.7	36.3	43.5	47.7
Zhu et al. [45]	19.0	28.9	36.2	43.6	47.7
SSAD	**24.6**	**35.0**	**43.0**	**47.8**	50.1

4.3 Comparison with state-of-the-art systems

Results on THUMOS 2014. To train action classifiers, we use full UCF-101 dataset. Instead of using one background category, here we form background categories using 81 action categories which are un-annotated in detection task. Using two-stream and C3D networks as action classifiers, the dimension of SAS features is 303.

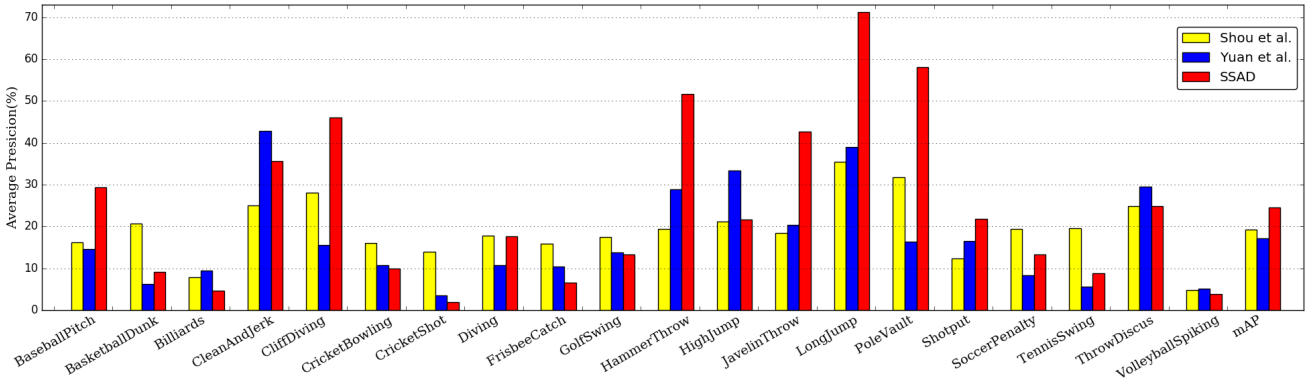

Figure 5: Detection AP over different action categories with overlap threshold 0.5 in THUMOS'14.

Table 2: Results on MEXaction2 dataset with overlap threshold 0.5. Results for [1] are taken from [29].

AP(%)	BullCHargeCape	HorseRiding	mAP(%)
DTF [1]	0.3	3.1	1.7
SCNN [29]	11.6	3.1	7.4
SSAD	**16.5**	**5.5**	**11.0**

Table 3: Comparisons between different action classifiers used in SSAD on THUMOS'14, where two-stream network includes both spatial and temporal networks.

Action Classifier used for SAS Feature	mAP ($\theta = 0.5$)
C3D Network	20.9
Two-Stream Network	21.9
Two-Stream Network+C3D Network	**24.6**

Table 4: Comparisons among multiple base layers configurations on THUMOS'14. A, B, C, D, E are base layers configurations which presented in Figure 3.

Network Configuration	A	B	C	D	E
mAP($\theta = 0.5$)	23.7	**24.6**	24.1	23.9	23.4

For training of SSAD model, we use 200 annotated untrimmed video in THUMOS'14 validation set as training set. The window length L_w is set to 512, which means approximately 20 seconds of video with 25 fps. This choice is based on the fact that 99.3% action instances in the training set have smaller length than 20 seconds. We train SSAD network for 30 epochs with learning rate of 0.0001.

The comparison results between our SSAD and other state-of-the-art systems are shown in Table 1 with multiple overlap IoU thresholds varied from 0.1 to 0.5. These results show that SSAD significantly outperforms the compared state-of-the-art methods. While the IoU threshold used in evaluation is set to 0.5, our SSAD network improves the state-of-the-art mAP result from 19.0% to 24.6%. The Average Precision (AP) results of all categories with overlap threshold 0.5 are shown in Figure 5, the SSAD network outperforms other state-of-the-art methods for 7 out of 20 action categories. Qualitative results are shown in Figure 6.

Results on MEXaction2. For training of action classifiers, we use all 1336 trimmed video clips in training set. And we randomly sample 1300 background video clips in untrimmed training videos. The prediction categories of action classifiers are "HorseRiding", "BullChargeCape" and "Background". So the dimension of SAS features equals to 9 in MEXaction2.

For SSAD model, we use all 38 untrimmed video in MEXaction2 training set training set. Since the distribution of action instances' length in MEXaction2 is similar with THUMOS'14, we also set the interval of snippets to zero and the window length T_w to 512. We train all layers of SSAD for 10 epochs with learning rate of 0.0001.

We compare SSAD with SCNN [29] and typical dense trajectory features (DTF) based method [1]. Both results are provided by [29]. Comparison results are shown in Table 2, our SSAD network achieve significant performance gain in all action categories of MEXaction2 and the mAP is increased from 7.4% to 11.0% with overlap threshold 0.5. Figure 6 shows the visualization of prediction results for two action categories respectively.

4.4 Model Analysis

We evaluate SSAD network with different variants in THUMOS'14 to study their effects, including action classifiers, architectures of SSAD network and post-processing strategy.

Action classifiers. Action classifiers are used to extract SAS feature. To study the contribution of different action classifiers, we evaluate them individually and coherently with IoU threshold 0.5. As shown in Table 3, two-stream networks show better performance than C3D network and the combination of two-stream and C3D network lead to the best performance. In action recognition task such as UCF101, two-stream network [40] achieve 91.4%, which is better than 85.2% of C3D [36] network (without combining with other method such as iDT [38]). So two-stream network can predict action categories more precisely than C3D in snippet-level, which leads to a better performance of the SSAD network. Furthermore, the SAS feature extracted by two-stream network and C3D network are complementary and can achieve better result if used together.

Figure 6: Visualization of prediction action instances by SSAD network. Figure (a) shows prediction results for two action categories in THUMOS'14 dataset. Figure (b) shows prediction results for two action categories in MEXaction2 dataset.

Table 5: Evaluation on different post-processing strategy on THUMOS'14.

p_{class}	✓		✓		✓	✓
p_{sas}		✓	✓	✓		✓
p_{over}				✓	✓	✓
mAP ($\theta = 0.5$)	22.8	13.4	24.3	19.8	23.3	**24.6**

Architectures of SSAD network. In section 3.3, we discuss several architectures used for base network of SSAD. These architectures have same input and output size. So we can evaluate them fairly without other changes of SSAD. The comparison results are shown in Table 4. Architecture *B* achieves best performance among these configurations and is adopted for SSAD network. We can draw two conclusions from these results: (1) it is better to use max pooling layer instead of temporal convolutional layer to shorten the length of feature map; (2) convolutional layers with kernel size 9 have better performance than other sizes.

Post-processing strategy. We evaluate multiple post-processing strategies. These strategies differ in the way of late fusion to generate p_{final} and are shown in Table 5. For example, p_{class} is used for generate p_{final} if it is ticked in table. Evaluation results are shown in Table 5. For the categories score, we can find that p_{class} has better performance than $\tilde{p}_{sas}$. And using the multiplication factor p_{over} can further improve the performance. SSAD network achieves the best performance with the complete post-processing strategy.

5 CONCLUSION

In this paper, we propose the Single Shot Action Detector (SSAD) network for temporal action detection task. Our SSAD network drops the proposal generation step and can directly predict action instances in untrimmed video. Also, we have explored many configurations of SSAD network to make SSAD network work better for temporal action detection. When setting Intersection-over-Union threshold to 0.5 during evaluation, SSAD significantly outperforms other state-of-the-art systems by increasing mAP from 19.0% to 24.6% on THUMOS'14 and from 7.4% to 11.0% on MEXaction2. In our approach, we conduct feature extraction and action detection separately, which makes SSAD network can handle concise high-level features and be easily trained. A promising future direction is to combine feature extraction procedure and SSAD network together to form an end-to-end framework, so that the whole framework can be trained from raw video directly.

REFERENCES

[1] 2015. MEXaction2. http://mexculture.cnam.fr/xwiki/bin/view/Datasets/Mex+action+dataset. (2015).

[2] M. Abadi, A. Agarwal, P. Barham, and others. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[3] F. Caba Heilbron, J. Carlos Niebles, and B. Ghanem. 2016. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1914–1923.

[4] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Feifei. 2009. ImageNet: A large-scale hierarchical image database. (2009), 248–255.

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

[6] C. Feichtenhofer, A. Pinz, and A. Zisserman. 2016. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1933–1941.

[7] J. Gemert, M. Jain, E. Gati, C. G. Snoek, and others. 2015. *Apt: Action localization proposals from dense trajectories*. BMVA Press.

[8] R. Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*. 1440–1448.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 580–587.

[10] X. Glorot and Y. Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks.. In *Aistats*, Vol. 9. 249–256.

[11] X. Glorot, A. Bordes, and Y. Bengio. 2011. Deep Sparse Rectifier Neural Networks.. In *Aistats*, Vol. 15. 275.

[12] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 675–678.

[14] Y. G. Jiang, J. Liu, A. R. Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. 2014. THUMOS challenge: Action recognition with a large number of classes. In *ECCV Workshop*.

[15] S. Karaman, L. Seidenari, and A. Del Bimbo. 2014. Fast saliency based pooling of fisher encoded dense trajectories. In *ECCV THUMOS Workshop*, Vol. 1.

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

[17] D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[18] H. Kuehne, H. Jhuang, R. Stiefelhagen, and T. Serre. 2013. HMDB51: A large video database for human motion recognition. In *High Performance Computing in Science and Engineering '12*. Springer, 571–582.

[19] C. Lea, R. Vidal, A. Reiter, and G. D. Hager. 2016. Temporal Convolutional Networks: A Unified Approach to Action Segmentation. In *Computer Vision–ECCV 2016 Workshops*. Springer, 47–54.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg. 2016. SSD: Single shot multibox detector. In *European Conference on Computer Vision*. Springer, 21–37.

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

[22] P. Mettes, J. C. van Gemert, and C. G. Snoek. 2016. Spot on: Action localization from pointly-supervised proposals. In *European Conference on Computer Vision*. Springer, 437–453.

[23] D. Oneata, J. Verbeek, and C. Schmid. 2014. The LEAR submission at Thumos 2014. *ECCV THUMOS Workshop* (2014).

[24] Z. Qiu, T. Yao, and T. Mei. 2016. Deep Quantization: Encoding Convolutional Activations with Deep Generative Model. *arXiv preprint arXiv:1611.09502* (2016).

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 779–788.

[26] J. Redmon and A. Farhadi. 2016. YOLO9000: Better, Faster, Stronger. *arXiv preprint arXiv:1612.08242* (2016).

[27] S. Ren, K. He, R. Girshick, and J. Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.

[28] A. Richard and J. Gall. 2016. Temporal action detection using a statistical language model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3131–3140.

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

[30] K. Simonyan and A. Zisserman. 2014. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*. 568–576.

[31] K. Simonyan and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

[32] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao. 2016. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1961–1970.

[33] G. Singh and F. Cuzzolin. 2016. Untrimmed Video Classification for Activity Detection: submission to ActivityNet Challenge. *arXiv preprint arXiv:1607.01979* (2016).

[34] K. Soomro, A. R. Zamir, and M. Shah. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402* (2012).

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

[36] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 4489–4497.

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

[38] H. Wang and C. Schmid. 2013. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision*. 3551–3558.

[39] L. Wang, Y. Qiao, and X. Tang. 2014. Action recognition and detection by combining motion and appearance features. *THUMOS14 Action Recognition Challenge* 1 (2014), 2.

[40] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. 2015. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159* (2015).

[41] R. Wang and D. Tao. 2016. UTS at activitynet 2016. *AcitivityNet Large Scale Activity Recognition Challenge* 2016 (2016), 8.

[42] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. 2016. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2678–2687.

[43] G. Yu and J. Yuan. 2015. Fast action proposals for human action detection and search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1302–1311.

[44] J. Yuan, B. Ni, X. Yang, and A. A. Kassim. 2016. Temporal Action Localization with Pyramid of Score Distribution Features. In *IEEE Conference on Computer Vision and Pattern Recognition*. 3093–3102.

[45] Y. Zhu and S. Newsam. 2016. Efficient Action Detection in Untrimmed Videos via Multi-Task Learning. *arXiv preprint arXiv:1612.07403* (2016).