

# Temporal Action Localization in Untrimmed Videos using Action Pattern Trees

Hao Song, Xinxiao Wu, Member, IEEE, Bing Zhu, Yuwei Wu, Mei Chen, Member, IEEE, and Yunde Jia, Member, IEEE

**Abstract**—In this paper, we present a novel framework of automatically localizing action instances based on action pattern trees (AP-Trees) in a long untrimmed video. For localizing action instances in varied temporal length videos, we first split videos into sequential segments and then use the AP-Trees to produce precise temporal boundaries of action instances. The AP-Trees can exploit the temporal information between segments of videos based on the label vectors of segments, by learning the occurrence frequency and order of segments. In AP-Trees, nodes stand for action class labels of segments and edges represent the temporal relationships between two consecutive segments. Thus we can discover the occurrence frequencies of segments by searching paths of AP-Trees. In order to obtain accurate labels of video segments, we introduce deep neural networks to annotate the segments by simultaneously leveraging the spatio-temporal information and the high-level semantic feature of segments. In the networks, informative action maps are generated by a global average pooling layer to retain the spatio-temporal information of segments. An overlap loss function is employed to further improve the precision of label vectors of segments by considering the temporal overlap between segments and the ground truth. The experiments on THUMOS2014, MSR ActionII, and MPII Cooking datasets demonstrate the effectiveness of our method.

**Index Terms**—Temporal Action Localization, Action Pattern Tree, Informative Action Maps, Overlap Loss Function.

## I. INTRODUCTION

Human action understanding has been an active topic in computer vision with many applications including video surveillance, human computer interaction, sports video analysis, and video retrieval [41], [38], [15], [7], [47]. Automatically discovering the temporal boundaries of action instances plays an important role in analyzing and understanding videos [54], [45]. In long untrimmed videos, there are typically a large number of background segments (not containing actions) that might degrade the performance of action localization, and there may be one or more action instances which occur at different time stamps in a video. Therefore, temporal action localization remains a challenging research problem particularly in determining “what action occurs, and when?”

To automatically localize action instances in long untrimmed videos, Shou *et al.* [32] and Zhu *et al.* [55] proposed to uniformly sample 16 frames from videos of different temporal scales. For long videos which may contain hundreds

Hao Song, Xinxiao Wu, Bing Zhu, Yuwei Wu, and Yunde Jia are with the Beijing Lab of Intelligent Information Technology, the School of Computer Science, Beijing Institute of Technology, Beijing 100081, China. Email: songhao, wuxinxiao, wuyuwei, jiayunde@bit.edu.cn. Mei Chen is with the Department of Electrical and Computer Engineering, State University of New York at Albany, NY 12222, USA. Email: meichen@albany.edu. Xinxiao Wu is the corresponding author.

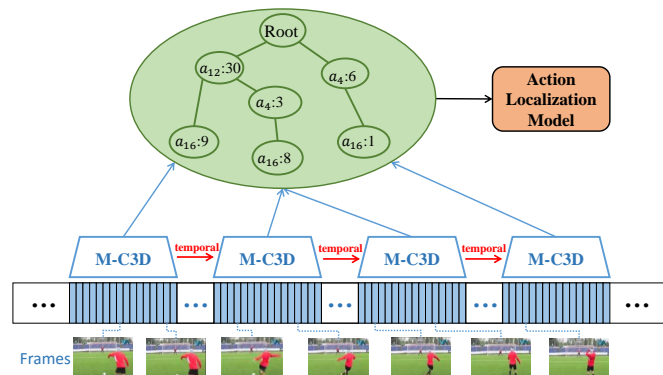


Fig. 1. An AP-Tree is built by using the temporal information of sequential segments. The video is split into consecutive segments with equal temporal length and each segment is recognized by the modified C3D model. The generated AP-Tree can exploit the temporal relationships between segments and learn the action localization model to produce precise localization boundaries, where  $Root$ ,  $a_4$ ,  $a_{12}$  and  $a_{16}$  are the elements of the AP-Tree, and the numbers indicate the occurrence frequency of these elements in training videos.

of frames, just uniformly selecting 16 frames cannot represent the video effectively, and the temporal information between consecutive frames was rarely considered. In our work, all the frames are utilized for temporal action localization, and each video is split into sequential segments with equal temporal length.

Due to the complexity of untrimmed videos, simple combination with the recognition results of these segments cannot produce accurate boundaries because each segment may be identified as different categories despite they belong to the same video. Therefore, we introduce an action pattern tree (AP-Tree) to exploit the temporal relationship between sequential segments in the entire video and aggregate the segments into complete action instances to produce more precise boundaries, as shown in Figure 1. The action class labels of segments are stored in the nodes and the temporal orders of consecutive segments are memorized in the edges of AP-Trees. The occurrence frequencies and orders of segments can be discovered by searching the paths of AP-Trees. For each action category, an AP-Tree is constructed by training videos from the corresponding action category. In testing, we predict the action label of the entire video by computing the relevances between the testing segments and all the trained AP-Trees. The AP-Tree can refine the temporal boundaries of action instances. If the segments in the test video contain

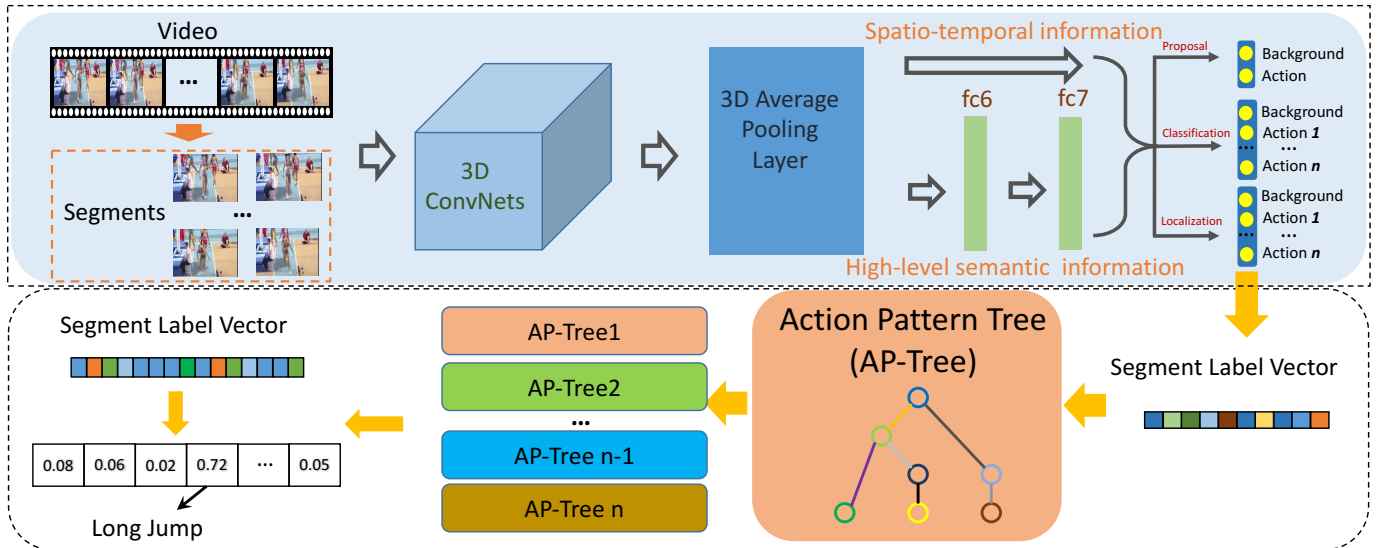


Fig. 2. The architecture of our framework. The top shows the details of the networks. The networks are based on the C3D model followed by three stage networks (“Proposal”, “Classification”, and “Localization”). The networks can label each segment in videos with varied temporal length and produce corresponding segment label vectors. Action pattern tree belonging to one action category is described in the middle, which models the occurrence order and frequency of segments by using label vectors of training videos. We use the generated AP-Trees to classify and localize the interested action instances in videos with varied temporal length. At last, the prediction scores of the test video corresponding to all the action classes are calculated and we select the index of the maximum value as the final recognition result.

the action items that did not appear in the corresponding AP-Trees, we will remove the segments to produce more precise temporal boundaries of action instances. To the best of our knowledge, our work is the first to employ AP-Trees for temporal action localization. The inputs of AP-Trees are label vectors of segments and we use deep neural networks to recognize segments accurately.

Owing to the advances of deep learning on video understanding [33], [51], [2], [19], inspired by [55], we employ the 3D ConvNets [35] to effectively recognize segments by simultaneously using the spatio-temporal information and the high-level semantic features. Specifically, the networks are trained to learn spatio-temporal action feature maps (the last convolutional layer **conv5**), which convolves the original videos on both the spatial and temporal dimensions. The feature map contains detailed temporal information, which significantly benefits representing actions by capturing the intrinsic motions. Moreover, the high-level feature (the fully connected layer **fc7**) is learned to describe actions at the semantic level which further improves the performance. Different from [55], we replace the last global max pooling layer with the global average pooling layer so that our method can generate more informative action maps with the **conv5** layer and learn high-level semantic features with the **fc7** layer to classify segments.

The networks for segment recognition are shown in the top of Figure 2. Based on the network proposed by Shou *et al.* [32], the networks are designed using three highly related 3D ConvNets. “Proposal” is a binary network to identify segments in a video that may contain actions. “Classification” is a multi-class classifier to group segments into different action categories and the background. “Localization” is a fine-tuned

classification network with a temporal overlap loss function which can produce more accurate categories of segments.

Our main contributions are three-fold:

- (1) We propose a novel framework to automatically localize action instances in long untrimmed videos, by capturing all the temporal information between consecutive frames as well as between sequential segments.
- (2) To the best of our knowledge, our work is the first to employ AP-Trees for temporal action localization in long untrimmed videos. The AP-Trees can learn the occurrence frequency and the order of sequential video segments in the temporal dimension, and produce precise temporal boundaries of interested actions.
- (3) We present deep neural networks to simultaneously exploit the spatio-temporal information and the high-level semantic features for effectively representing video segments, which can further improve the recognition accuracy of segments.

## II. RELATED WORK

### A. Temporal Action Localization

Many researchers have concentrated on temporal action localization in long untrimmed videos. Karaman *et al.* [20] used FV encoding of improved dense trajectory (iDT) [39] with weighted saliency based pooling, and conducted late fusion with frame-level CNN features. Wang *et al.* [40] built a system on iDT with FV representation and frame-level CNN features, and performed postprocessing to refine the detection results. Gaidon *et al.* [11] proposed an atomic action units sequence model (ASM) to represent an action as a temporal sequence of histograms of atom-anchored visual features for action localization. Jain *et al.* [18] introduced a sampling strategy to produce tubelets with motion information from super-voxels

for action localization. Ma *et al.* [25] described hierarchical space-time segments as new representations for action recognition and localization. Heilbron *et al.* [5] introduced a sparse learning dictionary method to recover the temporal segments containing interested actions. These methods utilize hand-crafted features with encoding methods to determine the temporal boundaries of action instances.

As deep convolutional neural networks (CNN) [10] have demonstrated breakthrough performance for image feature extraction, more and more studies of temporal action localization focus on deep learning. Shou *et al.* [32] utilized 3D ConvNets [35] to design a multi-stage framework for temporal action localization, which explicitly took the temporal overlap into account. They also presented convolutional-de-convolutional networks to predict actions at the frame-level granularity later [31]. Yeung *et al.* [45] formulated the localization model as a recurrent neural network (RNN) based agent and used reinforcement learning [42] to learn the agent's decision policy. The fully end-to-end network takes a long video as input and outputs the temporal bounds of all action instances. Dai *et al.* [8] presented a temporal context network to produce action proposals, rank and classify the proposals for temporal localization of human activities. Zhao *et al.* [52] modeled each action instance with a temporal pyramid, and all the context information of action instances are leveraged sufficiently. Hou *et al.* [17] proposed a tube convolutional neural network to localize actions based on 3D convolution features. Zhu *et al.* [55] also combined the 3D ConvNets with multi-task learning. In contrast with these complicated networks, we utilize deep networks to both learn the spatio-temporal information and the high-level semantic features to effectively recognize segments in videos. More importantly, we introduce action pattern trees to model the temporal relationship between segments and infer precise temporal boundaries of action instances.

### B. Temporal Structure Modeling

Many approaches focused on temporal structure modeling of videos for action recognition [26], [46], [44]. Sun *et al.* [34] utilized noisy labeled web images to generate localized action frames in videos and explored the temporal information by long short-term memory (LSTM) networks [16]. Ma *et al.* [24] directly discovered ensembles of hierarchical spatio-temporal trees from training data for action recognition. Truyen *et al.* [36] presented a boosting algorithm with the dynamic conditional markov random field to handle hidden variables (missing labels) of actions which is particularly attractive for smarthouse domains. Borzeshi *et al.* [3] used graphs to represent the shapes of actors and converted the graph into a suitable feature vector. Liu *et al.* [23] utilized skeleton-based tree traversal technique to feed the structure of the skeleton data into a sequential LSTM and improved the design of spatio-temporal LSTM by adding the trust gate to deal with noisy input for 3D action recognition. Different from these structure modeling methods, inspired by frequent pattern trees [14], we introduce action pattern trees to learn the occurrence frequency and order of segments and produce

precise temporal boundaries of action instances. The action pattern trees can explore the relationships between segments in the temporal dimension.

### C. Frequent Pattern Tree

Han *et al.* [14] firstly introduced the frequent pattern tree structure for storing crucial information about mining frequent patterns in transaction and time-series databases. They also developed the FP-Growth algorithm for efficient and scalable mining on both long and short frequent patterns. Chang *et al.* [6] proposed an incremental data mining algorithm based on FP-Growth using the concept of heap tree to address the issue of incremental updating of frequent itemsets. Aditya and Pradana [1] leveraged the FP-Growth algorithm to find the customer buying habits on market basket in organic medicine store. Dharmaraajan and Dorairangaswamy [9] utilized the FP-Growth algorithm to classify user behavior in identifying the patterns of the browsing and navigation data of web users. Wang *et al.* [37] extended the FP-Growth algorithm to mine both positive and negative frequent patterns of people's real health examination information. The main difference between these methods and our method is that we apply the frequent pattern tree to the action localization in computer vision by modeling the temporal information underlying the action videos.

## III. ACTION PATTERN TREE

To utilize all frames in videos with varied temporal lengths for temporal action localization, we divide each video into several short segments with equal temporal length. We utilize a modified C3D ConvNets to label each segment with both the spatio-temporal information and the high-level semantic features.

Inspired by Frequent Pattern Tree [14] which is widely used for mining frequent patterns in data mining, we propose action pattern trees (AP-Trees) to learn the occurrence frequency and the temporal order of segments before aggregating the segments into complete action instances. In the training stage, we build an AP-Tree for each action category with training videos, and in the testing stage, we compute the similarity score between the test video and all the learned AP-Trees to determine the action category and the temporal boundaries of action instances within the video.

### A. The construction of Action Pattern Tree

After inferring the labels of segments in the training videos belonging to one action class, we utilize these label vectors to build its corresponding AP-Tree. Similar to the frequent pattern tree, AP-Tree stores sufficient information of videos. In the AP-Trees, each node represents the action class label of the segment, the edge indicates the occurrence temporal order between two consecutive segments, and the occurrence frequencies of the segments are recorded as the value of each node. In this way, each path of the AP-Trees can maintain the complete temporal information of a video.

Specifically, let  $A = \{a_1, a_2, \dots, a_N\}$  be the set of action category labels, where  $N$  is the number of action categories.

VID	Sequential Action Items
1	$a_1, a_3, a_2, a_4, a_5$
2	$a_1, a_3, a_2, a_6, a_4$
3	$a_1, a_6$
4	$a_3, a_4, a_5$
5	$a_1, a_3, a_2, a_4, a_5$

TABLE I

EXAMPLES OF ACTION ITEMS OF THE "BASEBALL PITCH" ACTION.  $a_i \in A$  ARE THE LABELS OF THE SEGMENTS. ALL THE ITEMS OCCURS ALONG WITH THE TEMPORAL DIMENSION.

The elements in  $A$  are called action items. The training dataset is indicated by  $T = \{v_1, v_2, \dots, v_m\}$  where  $m$  is the number of training videos and  $v_i$  is a training video which contains a set of action items. A pattern is defined by a set of action items. The support (i.e. occurrence frequency) of a pattern  $P$  is the number of training videos containing  $P$ .  $p$  is defined as an action pattern if the support of  $p$  is no less than a predefined threshold  $\xi$ .

With the label vectors of segments in training videos, the action item (i.e., action class label) with the highest occurrence frequency is treated as the main pattern of the AP-Tree. Different from the construction of frequent pattern tree, we construct the action pattern tree by extending the main action pattern along the temporal dimension. The action items after the main action item are added to construct the left branches of the AP-Tree and the action items before the main action item are utilized to construct the right branches. The same action items in an AP-Tree are accumulated and the different action items produce the sub-action nodes based on the parent-action node. The detailed algorithm for constructing an AP-Tree is shown in Algorithm 1. Each path from the root node to the sub-node in the AP-Tree is an action pattern and the number of action patterns is also counted in the construction process to represent its occurrence frequency.

Let us take the action class of "Baseball Pitch" as an example. Suppose that the training dataset  $T$  contains 5 training videos, and the action class label vectors of the sequential segments in these videos are shown in Table I. Then, an action pattern tree is constructed as follows. We first scan the training set to obtain a set of action items  $\{a_1:4, a_3:4, a_2:3, a_4:3, a_5:3, a_6:3\}$  (the number indicates the support) in which the action items are ordered in descending frequency. Each path of a tree will also follow this order. Then, we create the root of the tree, labeled as "Root". Again we scan the training dataset. With the first training video, we construct the first branch of its action pattern tree:  $\{a_1:1, a_3:1, a_2:1, a_4:1, a_5:1\}$ . The action items are ordered according to the temporal dimension. For the second training video, its  $\{a_1, a_3, a_2, a_6, a_4\}$  shares a common prefix  $\{a_1, a_3, a_2\}$ . The count of each node along the prefix is incremented by 1, one new node  $\{a_6:1\}$  is created and linked as a child of  $\{a_2:2\}$  and another new node  $\{a_4:1\}$  is created and linked as the child of  $\{a_6:1\}$ . For the third video, as the rule mentioned above, the count of  $a_1$  is added by 1, and a new node  $\{a_6:1\}$  is created and linked to  $\{a_1:3\}$ . The scan of the fourth video leads to constructing the second branch of the action pattern tree. For the last video, since it is the same as the first video, the count of each node along the path should

### Algorithm 1 Pseudo-code of the Construction of Action Pattern Tree (AP-Tree)

**Require:** The label vectors of the training set  $T$  on one action category, the minimum action item support  $\xi_1$ .

**Ensure:** An Action Pattern Tree of  $R$ .

- 1: State all the action items  $A$  from  $T$ , and the *support* of them.
- 2: Filter  $A$  with  $\xi_1$ , and produce action item list  $L$  by sorting them in descending order with their *support*.
- 3: Create the root node in the AP-Tree, label it as "Root". Label the first action item  $ap$  in  $L$  as the left sub-node of "Root".
- 4: Scan the pattern set  $T$ .
- 5: **FOR each**  $v$  **in**  $T$
- 6:     Find the main action item  $a$  and stat the count  $c$ .
- 7:     **FOR number from** 1 **to**  $c$
- 8:     **IF an action item**  $a'$  **occurs before**  $a$
- 9:         **IF**  $a$  **has a left child node**  $a'$
- 10:             the count of  $a'$  adds 1.
- 11:         **ELSE**
- 12:             Create the new left child  $a'$  of  $a$  and set the count as 1.
- 13:         **ELSE an action item**  $a'$  **happens after**  $a$
- 14:             **IF**  $a$  **has a right child node**  $a'$
- 15:                 the count of  $a'$  adds 1.
- 16:             **ELSE**
- 17:                 Create the new right child  $a'$  of  $a$  and set the count as 1.
- 18:     **END**

be incremented by 1. Each path in the action pattern tree is treated as one action pattern of this class when its support (count) is larger than a threshold.

The AP-Tree is a compact and compressed data structure which contains all action pattern information in training videos of one action category. The depth of the AP-Tree represents the complexity of action items. The width of the AP-Tree is determined by the action patterns and illustrates the types of action patterns around the main action pattern. Each branch of the AP-Tree holds the temporal information of segments in videos. Due to the fact that there is a large number of shared action patterns in the training videos, the size of a typical AP-Tree is smaller than the basic action pattern set. One example of an action pattern tree is shown in Figure 3.

### B. Evaluation of videos with AP-Trees

Each video in the training dataset can be mapped to one path in the AP-Tree. The nodes and edges of the AP-Tree store action items and temporal information of the segments, respectively. Moreover, one path in the AP-Tree can represent the frequent action items in multiple videos without ambiguity. We measure the similarity between the input test video  $v_i$  and each AP-Tree by calculating the relevance between the sequential action items in the video  $v_i$  and the action patterns  $p_i$  in each AP-Tree. Then, the score of the video  $v_i$  is

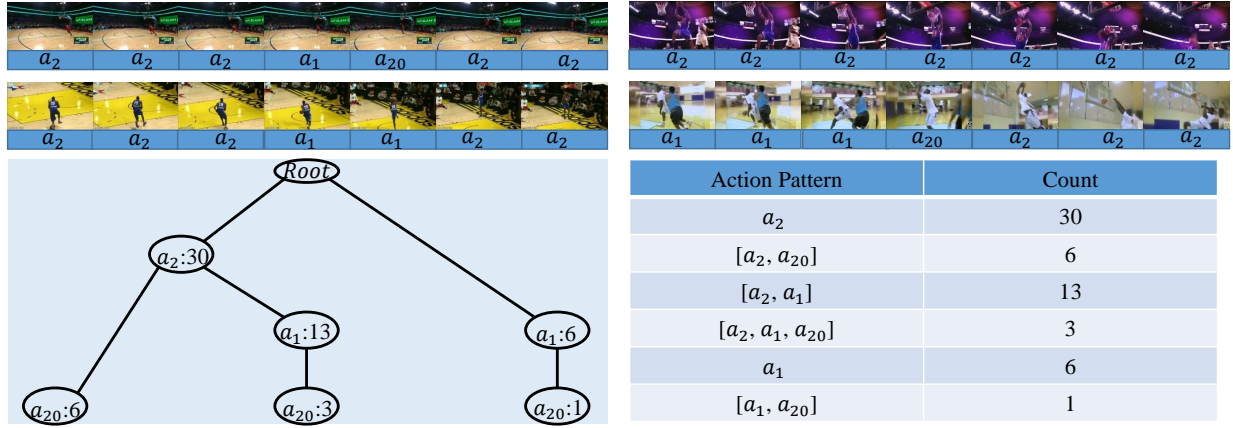


Fig. 3. The AP-Tree of the “BasketballDunk” action. The top two rows are four sampled training videos of the action “BasketballDunk”. The frame stands for the first frame of the segments in the video. We first label segments in training videos by the proposed neural network and then build an AP-Tree by the label vectors. The Left part in the below is the AP-Tree we constructed. The right portion is the action patterns and their support of this AP-Tree. For example, the action items occur in “BasketballDunk” are  $\{a_1, a_2, a_{20}\}$ , the action patterns  $\{a_2\}$ ,  $\{a_2, a_1\}$ ,  $\{a_2, a_{20}\}$  are the frequent action patterns we collect from the training samples, which represent the consecutive segments with high occurrence frequency.

computed by

$$S(v_i) = \sum_{i=1}^{N_1} \alpha_i \cdot Num(a_i) + \sum_{i=1}^{N_2} \beta_i \cdot Num(p_i) + F, \quad (1)$$

where  $Num(a_i)$  indicates the number of the action item  $a_i$  in the video  $v_i$ .  $Num(p_i)$  denotes the number of the action pattern  $p_i$  in the AP-Tree.  $N_1$  and  $N_2$  are the total number of action items and action patterns in the AP-Tree, respectively.  $F$  is a penalty term defined as  $F = \sum_{j=1}^{N_3} \lambda_j$ , where  $N_3$  is the number of action items occurring in the video  $v_i$ , and  $\lambda$  is set to negative as a penalty coefficient when the action item  $a_j$  does not exist in the AP-Tree, otherwise is set to zero.  $\alpha_i$  and  $\beta_i$  are the weights of  $a_i$  and  $p_i$ , respectively, defined by

$$\alpha_i = \frac{C(a_i)}{\sum_{j=1}^N C(a_j)}, \beta_i = \frac{C(p_i)}{\sum_{j=1}^N C(p_j)}, \quad (2)$$

where  $C(a_i)$  and  $C(p_i)$  are the numbers of the action item  $a_i$  and the action pattern  $p_i$ , respectively.

Specifically, the penalty term  $F$  will punish the action term which does not occur in the AP-Tree. With the score generated by the action pattern tree, we can determine the action category of the test video. Additionally if one action item does not occur in the action pattern tree, the corresponding segment will be discarded in the determination process of temporal boundaries. In this way, the AP-Tree can also contribute to refining the temporal boundaries of action instances in long untrimmed videos.

#### IV. NEURAL NETWORK ARCHITECTURE

In a long untrimmed video, an action instance could occur at different time stamps and may last for a few seconds. In order to produce precise labels of segments in videos, we use deep neural networks to describe segments by exploiting the spatio-temporal information and the high-level semantic feature.

We base our network on the C3D neural network [35] which conducts convolutions in both spatial and temporal dimensions. Specifically, we propose to leverage the informative

action maps (the last convolutional layer **conv5**) which contain sufficient spatial and temporal information of input segments and the high-level features (the fully connected layer **fc7**) that represent input segments at the semantic level.

#### A. Informative Action Maps

Inspired by the work of Zhou *et al.* [53], we utilize informative action maps to recognize segments in videos. The action maps maintain the characteristics of actions and capture 3D discriminative action regions to identify the action in a specified category. Figure 4 shows the network architecture of generating the informative action maps. Specifically, we remove all the fully-connected layers and the last max pooling layer (i.e. **pool5**) in the C3D network. And we perform the Global Average Pooling (GAP) on the 3D convolutional feature maps with a fully-connected layer as the desired output of the learned network. In our work, we propose to replace the Global Max Pooling (GMP) layer with the GAP layer. When operating the GAP of feature maps produced by the last convolutional layer, all the values of the feature maps are collected to find more discriminative action regions because the low values will make contributions to the final output. On the contrary, GMP ignores the low values in each feature map. From the experimental results, GAP achieves better performance than GMP.

For an input segment, let  $r_k(x, y, z)$  represent the response of unit  $k$  in the last convolutional layer (**conv5b**) at the spatio-temporal location  $(x, y, z)$ . With global average pooling operation  $f$ , the output of unit  $k$  can be computed by  $f_k = \sum_{x,y,z} r_k(x, y, z)$ . Thus, the input of the softmax layer for the action category  $c$  can be calculated by  $S_c = \sum_k w_k^c f_k$ , where  $w_k^c$  is the weight of class  $c$  for the unit  $k$ . The output of the softmax layer is given by  $\frac{\exp(S_c)}{\sum_c \exp(S_c)}$ .

Thus, we have

$$S_c = \sum_k w_k^c \sum_{x,y,z} r_k(x, y, z) = \sum_{x,y,z} \sum_k w_k^c r_k(x, y, z). \quad (3)$$



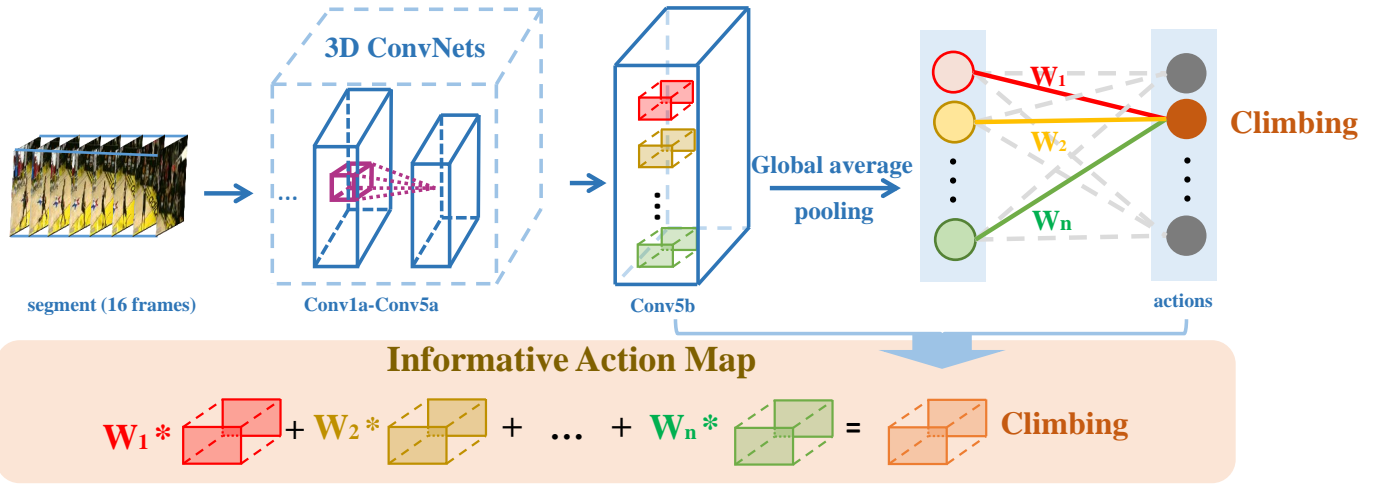


Fig. 4. The network architecture of producing informative action map. With this network, we can utilize the spatio-temporal information to generate specific action labels. The input of the network are short segments (16 consecutive frames in our experiment). The 3D ConvNets is adopted as in the C3D network except for the max pooling5 layer. All three fully-connected layer are removed, instead, a global average pooling layer and a fully-connected output layer are incorporated. The bottom shows the feature maps (red, yellow, green) learned from the last convolutional layer **Conv5b**, and the orange one is the informative action map produced by projecting the weights of the output layer into the convolutional feature maps. This map is discriminative for labeling segments by leveraging the spatio-temporal information.

Let  $M_c(x, y, z)$  be the response map for the class  $c$ :

$$M_c(x, y, z) = \sum_k w_k^c r_k(x, y, z), \quad (4)$$

where  $M_c(x, y, z)$  represents the importance of the response at the spatio-temporal location  $(x, y, z)$  in the last convolution layer **conv5b** for the classification of a video to the class  $c$ . Then,  $S_c$  can be given by  $S_c = \sum_{x,y,z} M_c(x, y, z)$ , which indicates that the learned response map contains sufficient information for classification.

### B. High-level Semantic Feature

In order to enhance the accuracy of classifying segments in videos, we also propose to extract high-level semantic features of segments. In contrast with the action maps which contain 3D action regions, the fully connected layer (**fc7**) of the 3D ConvNets contains the high-level semantic information of segments by maintaining the significance of segments.

In order to further improve the recognition accuracy of video segments, we combine the spatio-temporal information (the last conv layer **conv5**) and the high-level semantic information (the fully connected layer **fc7**) for recognition. As shown in Figure 2, two network branches are used to extract the spatio-temporal information and the high-level semantic information, respectively. Then, these two kinds of information are combined by using the average pooling operation on the two classification scores produced by the two branches.

### C. Three Related 3D ConvNets

Based on the networks proposed by Shou *et al.* [32], to classify the video segments, three related 3D ConvNets, including the Proposal network, Classification network and Localization network are employed. The Proposal network identifies segments that may contain actions in a video. The

Classification network can be treated as a multi-class classifier with  $N + 1$  classes ( $N$  action categories and the background). The Localization network explicitly takes the temporal overlap of segments into account and produces more accurate labels of segments.

In the Proposal network, the output of the network are two categories: “Action” and “Background”. In the training process, the segments from the trimmed training videos are all treated as positive. For the untrimmed videos, we only select the segments included in the ground truth as positive. The negative examples are background segments, which are randomly sampled from background videos. The number of training examples from both background and action classes are guaranteed to be equal.

After eliminating background segments, we train the Classification network for  $N + 1$  classes. The loss function is the conventional softmax loss function. In the training process, we assign labels of positive segments from each action category. In order to balance the number of training data for each action category, the number of the background samples is reduced to the mean of the numbers of action training examples.

As shown in Figure 5, several segments which have small overlaps with the ground truth also have high prediction scores. This will degrade the estimation performance of labeling segments. The temporal overlap of segments influences the localization results. In the Localization network, we use an overlap loss function, in which the prediction scores of segments with large overlap with the ground truth will be increased and the scores of segments with small overlap will be decreased.

We first define the overlap of a segment as  $ov$ . In the training data,  $ov$  of each segment in the trimmed videos is set to 1, and it is defined as the real overlap (measured by Intersection over Union, IoU) with the ground truth in untrimmed videos.

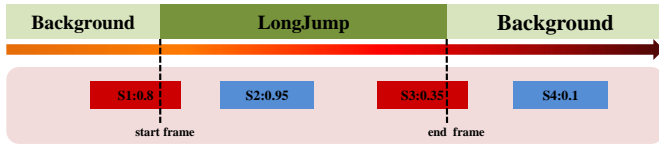


Fig. 5. The prediction scores of the segments in a long untrimmed video. The red segment  $S1$  has small overlap with the ground truth, but it has high prediction score, and the red segment  $S3$  has high overlap with the ground truth, but it achieves bad result. The scores of these segments should be refined with the “Localization” network. For the prediction score of the two blue segments  $S2$  and  $S4$ , these results should be maintained in the “Localization” network.

In the training step, we input  $N$  samples in each mini-batch. For the  $n$ -th training sample, the output of the last layer is  $O_n$  and the prediction score of the softmax layer is  $S_n$ . For the  $c$ -th class, the prediction score is calculated by  $s_n^c = \frac{e^{O_n^c}}{\sum_{j=1}^N e^{O_n^j}}$ . Then the overlap loss function can be formulated as

$$L_{overlap} = \frac{1}{N} \sum_n \left( \frac{S_n^{(l_n)}}{ov_n^2} - 1 \right) \cdot [l_n > 0], \quad (5)$$

where  $[\cdot]$  is a binary function.  $[l_n > 0] = 1$  when the true class label  $l_n$  is positive, otherwise  $[l_n > 0]$  is equal to 0 when the training segment is background. The gradient of the output of the  $i$ -th node in the last layer is computed by

$$\frac{\partial L_{overlap}}{\partial O_n^{(i)}} = \begin{cases} \frac{1}{N} \cdot \left( \frac{S_n^{(l_n)}}{ov_n^2} \cdot (1 - S_n^{(l_n)}) \right) \cdot [l_n > 0], & \text{if } i = l_n, \\ \frac{1}{N} \cdot \left( \frac{S_n^{(l_n)}}{ov_n^2} \cdot (-S_n^{(i)}) \right) \cdot [l_n > 0], & \text{if } i \neq l_n. \end{cases} \quad (6)$$

This overlap loss function is sensitive to the overlap of the segment with the ground truth. When the overlap  $ov = 1$ , the overlap loss function equals to the softmax loss function of the Classification network. Our overlap loss is similar to the loss in the work [32]. Different from [32], our overlap loss is computed by the inverse of square operation of the overlap. Once the overlap decreases, the loss will increase more heavily than the operation in [32]. In the training step, we finetune the network with the overlap loss function on the “Classification” network. The detailed parameter settings are given in Section V. The networks which exploit the spatio-temporal and high-level semantic information of videos are trained respectively.

## V. EXPERIMENT

### A. Dataset

**THUMOS 2014 dataset.** The THUMOS2014 dataset is a large action dataset in terms of number of classes, length and number of videos, which contains over 25M frames and over 254 hours of video data. It provides four separate subsets, including the training, validation, background, and testing subsets. (1) The training subset includes all the UCF101 dataset. (2) The validation subset has 1,010 temporally videos which are not trimmed. In general, each video has one action class. However, some videos may include one or more different actions. Each action class contains about 10 positive examples. (3) The background subset contains over 2,500 relevant videos which are ensured that they do not include any instance of the

101 action classes. Each background video corresponds to only one action class. (4) The test subset contains 1,574 temporally untrimmed videos with ground truth. Some videos may involve more than one action class. The action occurs at different time stamps in the video. In the temporal action localization stage, only 213 videos which contain interesting action instances are included. In the experiment, we use the videos in the validation set to finetune the basic deep neural networks. The UCF101 dataset is an action recognition dataset containing 13,320 videos from YouTube with 101 action classes. Each class contains more than 100 video clips, and all of which are temporally trimmed. The dataset has large variations in camera motion, appearance, scale, viewpoint, and cluttered background. It also exhibits a lot of diversity in terms of actions.

**MSR Action Dataset II.** The MSR Action II Dataset [49] consists of 54 video sequences with 203 action instances recorded in a crowded environment. There are three action types: hand waving, handclapping, and boxing. We follow the standard cross-dataset evaluation protocol and use the KTH dataset [30] for training.

**MPII Cooking.** The MPII Cooking dataset [29] is a large, fine-grained cooking activities dataset. It records 44 videos with a total length of more than 8 hours (881,755 frames) of 12 participants performing 65 different cooking activities, such as *cut slices*, *pour*, and *spice*. Following the standard protocol in [29], we have 7 splits after performing leave-one-person-out cross-validation. Each split uses 11 subjects for training, leaving one for validation. We list several sampled frames of the actions from these three datasets in Figure 6.

### B. Experiment Setup

The 3D ConvNets [19], [35] implement 3D convolutions on both spatial and temporal dimensions simultaneously, and capture both appearance and motion information of videos. Our neural network utilizes the 3D ConvNets as the basic architecture and follows the architecture of [35]. Our network has 8 convolutional layers, 5 pooling layers (4 max pooling layers and one averaging pooling layer), followed by two branches to leverage the spatio-temporal information and the high-level semantic feature for action localization. All the 3D convolutional filters are  $3 \times 3 \times 3$  with stride 1, and all the 3D pooling layers are  $2 \times 2 \times 2$  with stride 1, except for pool1 ( $1 \times 2 \times 2$ ) to preserve the temporal information in the earlier layers. We use the C3D network as the initialization for the Proposal and Classification network. The pre-trained C3D neural network is exploited in the experiment and each input of this network is a video clip with the size of  $171 \times 128 \times 16 \times 3$ . The basic model is trained on the largest video benchmark Sports-1M dataset [21].

During the finetuning process, the detailed settings of the networks for all the three datasets are listed as follows. We set the momentum to 0.9, and the weight decay factor to 0.0005. (1) For the THUMOS 2014 dataset, we perform 10K iterations with the learning rate of  $10^{-4}$ , and decrease the learning rate by  $\frac{1}{10}$  for every 10K iterations. (2) For the MSR Action dataset II, in the proposal network, we perform 8K



Fig. 6. The sampled frames of the action instances from three datasets. The action labels in each dataset are as follows. (1) THUMOS 2014 Dataset (complex): BaseballPitch, Billiards, CricketShot, Diving, FrisbeeCatch, GolfSwing, Hammer. (2) MSR ActionII Dataset (simple): Clapping, Handwaving, Boxing. (3) MPII Cooking Dataset (fine-grained): open/close fridge, cut apart, peel, wash objects.

iterations with  $10^{-4}$ , and then 8K iterations with  $10^{-5}$ , and 4K iterations with  $10^{-6}$  at last. In the classification network, we perform 4K iterations with the learning rate of  $10^{-4}$  and drop the learning rate by 10 every 4K iterations. (3) For the MPII Cooking dataset, we perform 10K iterations with  $10^{-4}$ , and drop the learning rate by 10 every 10K iterations. The number of total iterations is determined by the scale of the training dataset.

We use 70% of the training set to finetune the deep neural network proposed in this paper, and the left 30% training set is utilized to train the action pattern trees. The time complexity of training AP-Tree is  $O(n^2)$ , where  $n$  is the number of the training video samples. In the test procedure, the AP-Trees can deal with about 320 frames per second.

### C. Training and Post-processing

In this work, we apply a sliding window with 75% overlap to generate training and testing samples with multiple temporal scales. After producing these videos, we split each video into sequential segments with equal temporal length. The generated segments then are used to train all the deep neural networks. The trained networks are used to recognize the segments of the videos to generate the label vectors of training videos. After removing all the background segments, we build one AP-Tree for each action category with the refined training videos.

During the test process, we first classify the segments of testing videos with the networks, and compute the decision score of each generated test sample with action pattern trees. According the classification results of video samples, we remove the samples which are classified into the background category. Finally, we use the Non-maximum Suppression (NMS) method to generate the un-refined action instances from the video samples which are classified into action categories. After generating the un-refined action instances, if the action instance contains the action items which do not occur in the corresponding AP-tree, we will eliminate the segments with un-occurred action items to refine the temporal boundaries of action instances.

### D. Results on THUMOS Challenge 2014

1) *IoU of Action Temporal Localization*: Table II illustrates the comparison results of our method and several state-of-the-art methods on temporal action localization. As shown in

TABLE II  
TEMPORAL ACTION LOCALIZATION RESULTS ON THE THUMOS 2014 DATASET WITH VARIED IOU THRESHOLD  $\alpha$ . ALL THE RESULTS ARE SHOWN USING MAP.

	$\alpha$				
	0.1	0.2	0.3	0.4	0.5
<b>The Handcrafted Features</b>					
Karaman <i>et al.</i> [20]	1.5	0.9	0.5	0.3	0.2
Wang <i>et al.</i> [40]	19.2	17.8	14.6	12.1	8.5
Oneata <i>et al.</i> [27]	39.8	36.2	28.8	21.8	14.3
Heilbron <i>et al.</i> [5]	36.1	32.9	25.7	18.2	13.5
<b>Deep Neural Networks</b>					
Sun <i>et al.</i> [34]	12.4	11.0	8.5	5.2	4.4
Shou <i>et al.</i> [32]	47.7	43.5	36.3	28.7	19.0
Yeung <i>et al.</i> [45]	48.9	44.0	36.0	26.4	17.1
Zhu <i>et al.</i> [55]	47.7	43.6	36.2	28.9	19.0
Buch <i>et al.</i> [4]	-	-	-	-	23
Lin <i>et al.</i> [22]	50.1	47.8	43.0	35.0	24.6
Shou <i>et al.</i> [31]	-	-	40.1	29.4	23.3
Zhao <i>et al.</i> [52]	66.0	59.4	51.9	41.0	29.8
Gao <i>et al.</i> [12]	60.1	56.7	50.1	41.3	31.0
Xu <i>et al.</i> [43]	54.5	51.5	44.8	35.6	28.9
Yuan <i>et al.</i> [50]	51.0	45.2	36.5	27.8	17.8
Ours	48.5	44.1	38.2	29.8	20.1

Table II, our method outperforms other methods at most values of the overlap threshold  $\alpha$ , which indicates the effectiveness of our localization framework. Compared with the methods [20], [40], [27], [5] based on the handcrafted features, our method achieves better result which evaluates the effectiveness of utilizing both spatio-temporal and high-level semantic feature to represent segments with deep neural networks. In contrast to some deep network methods [34], [32], [45], [55], we introduce the AP-Trees to model the temporal relationship between segments and refine the temporal boundaries of the action instances, which can further improve the performance of action localization.

In Table II, our method achieves better results than the four methods at the top which used improved dense trajectory features with fisher vector (FV) encoding methods and post-processing to produce temporal boundaries of action instances. [27] is the winner of the THUMOS2014 challenge, which accomplishes localization using FV encoding of improved Dense Trajectory(IDT) feature on temporal sliding windows. In [27], the video-level classification scores greatly improves the results of localization. Our network could produce more accurate labels of segments in videos than the handcrafted



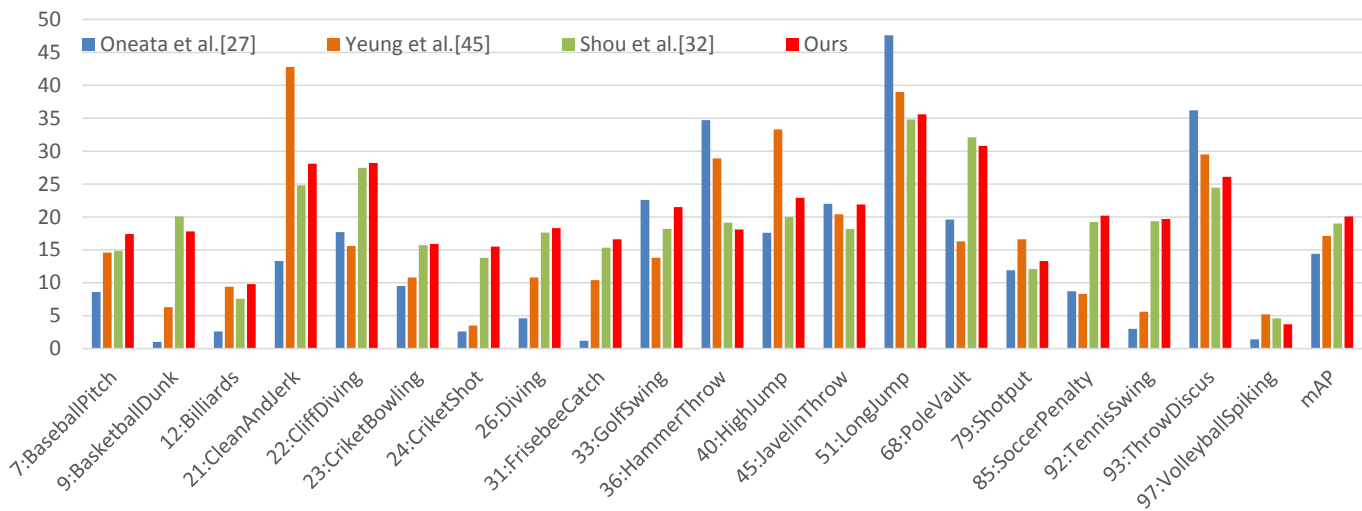


Fig. 7. Average precision (%) of temporal action localization between different methods on the THUMOS 2014 dataset when the overlap threshold is set to 0.5.

features. The approaches in the bottom are the state-of-the-art methods using deep neural networks. Yeung *et al.* [45] achieves better accuracy than our method in  $\alpha = 0.1$ , because the learned RNN network treats frames as the input. With reinforcement learning method, it tends to produce more temporal boundaries with small ranges. It under-performs our method when the test action instances are long. Compared with the methods [32] and [55], despite that we all utilize 3D ConvNets as the basic architecture, the results show that our method can achieve more accurate temporal boundaries due to the fact that we explore more temporal information of long videos through the AP-Trees.

As shown in Table II, our method performs a little worse than the methods [4], [22], [31], [52], [12], [43], [50]. Different from these methods, we just utilize a network to recognize the segments in videos and propose the AP-Trees algorithm to model the temporal relationship between these segments for temporal action localization. Buch *et al.* [4] proposed to use the C3D network and the recurrent GRU-based sequence encoder model for generation of action proposals, which can generate more accurate action proposals for temporal action localization. We just use simple sliding windows to produce action proposals. In the network [22], three anchor layers and convolutional layers are used to predict classification score and location offset of each video which is represented by snippet-level action score. Compared with [22], our method is relatively simple and just use networks in infer action labels of video segments. Shou *et al.* [31] designed convolutional-deconvolutional (CDC) filters on the top of 3D ConvNets to directly generate the action score of each frame, which can generate more precise temporal boundaries than our segment-based method. Different from the complicated network based methods [12], [43], [50], we just utilize networks to recognize the segments in videos and utilize the AP-Trees algorithm to model the temporal information between these segments for temporal action localization.

Compared with the complicated networks in [4], [22],

[31], [12], [43], [50], our AP-Tree method has following advantages: (1) The AP-Tree is a simple data structure which records the occurrence frequency and temporal relationships of segments in videos, without a large number of parameters which are required in networks. (2) Training an AP-Tree requires a relatively small amount of data compared with the large number of data for training networks.

2) *mAP of 20 Classes on the THUMOS 2014 dataset:* We compare our method with the state-of-the-art methods on the per-class AP on the THUMOS 2014 dataset in Figure 7. The overlap threshold  $\alpha$  is set to 0.5, and we follow the same experimental setting of  $\alpha$  as that in [32], [55]. As shown in Figure 7, our method obtains improvements on the challenging classes such as “CliffDiving”, “FrisbeeCatch”, and “SoccerPenalty”. Our network can learn more discriminative features to label segments of videos in these action categories. The AP-Trees can learn the occurrence frequency and the order of the segments, thereby producing more accurate temporal boundaries of action instances.

3) *Results of Temporal Localization on the THUMOS 2014 Dataset:* In Figure 8, we list the prediction results of five action instances on the THUMOS2014 test dataset. The action categories are “JavelinThrow”, “CliffDiving”, “LongJump”, “PoleVault”, “ThrowDiscus”, “Billiards” and “Shotput”. The figure shows the temporal boundaries of each action instance, and several frames are sampled from segments in the video to represent the entire action instance. For the “JavelinThrow”, the consecutive frames are similar, and the predicted temporal boundary is slightly larger than the ground truth. Due to the fact that the person appears later in the sequence, several initial frames of the ground truth in “CliffDiving” are labeled as background by our networks, and the temporal boundaries produced by our method are smaller. The temporal boundaries of the action “LongJump” are accurately predicted, which demonstrates the effectiveness of our localization framework.

Also, there are several negative detections, as shown in Figure 9. For the action “Billiards”, despite the detection results

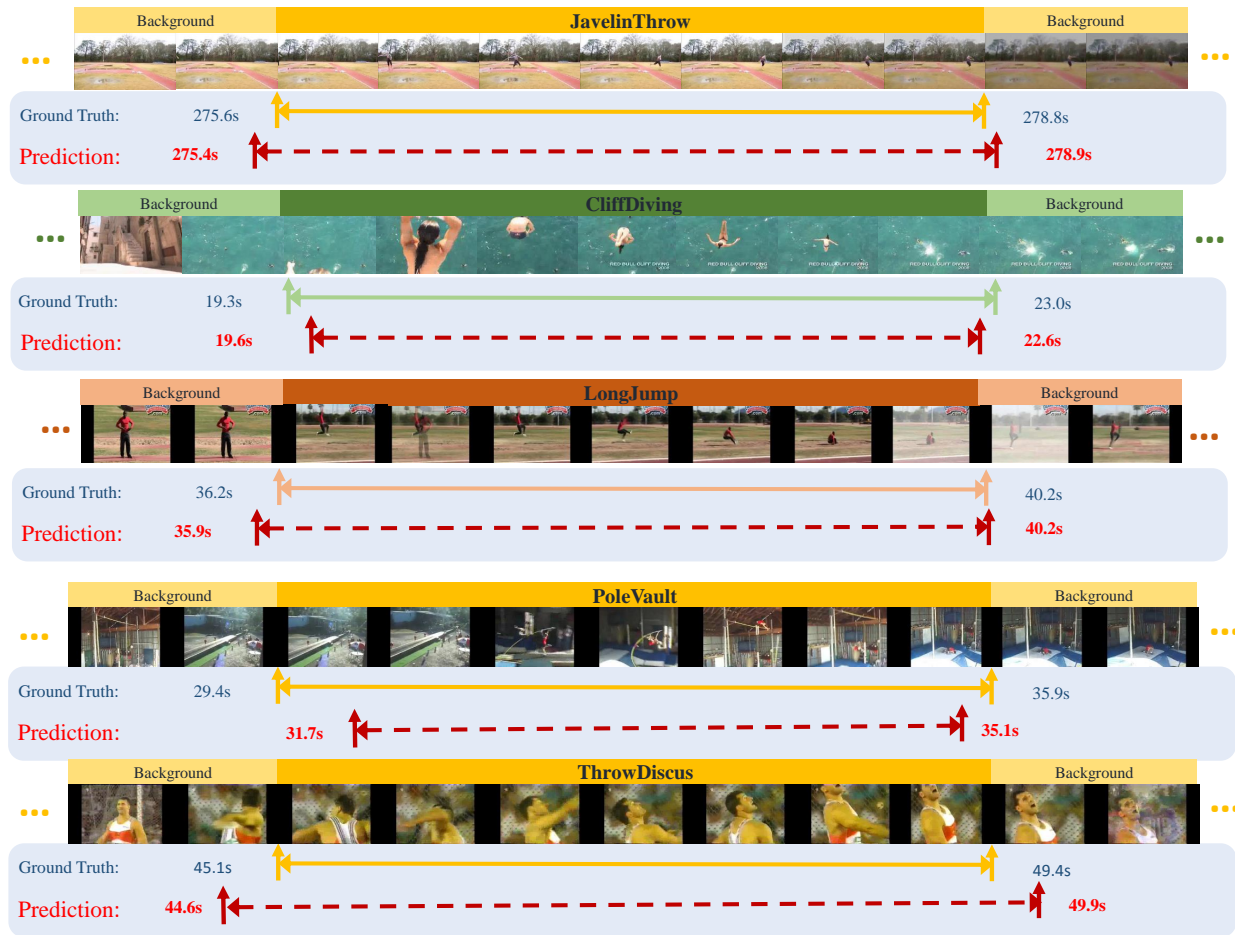


Fig. 8. The prediction results of five action instances (“JavelinThrow”, “CliffDiving”, “LongJump”, “PoleVault” and “ThrowDiscus”) on the THUMOS2014 dataset. We display the ground truth and the prediction results in the gray section.

of temporal boundaries are correct, the action “Billiards” is wrongly classified into “CricketBowling”. The reason may be that the spatio-temporal information of these two categories (i.e., “Billiards” and “CricketBowling”) are so similar that our network can not distinguish them when classifying the video segments. For the action “Shotput”, our method yields the wrong temporal boundary of the action instance. The reason may be that the frames around the ground truth are similar although they belong to different categories (i.e., background and action). The segments consisting of these similar frames are easily recognized as the same action category. For the action “LongJump”, we predict the video as background due to the complex scenes.

### E. Results on the MSR ActionII Dataset

Table III shows that our localization framework performs at least 2% better than the best of the state-of-the-art methods. Different from [5], our network can learn discriminative features which retain spatio-temporal information and high-level semantic information than the improved dense trajectory feature. Compared with [55], the proposed AP-Trees can utilize all frames of videos and explore sufficient temporal information between consecutive segments for predicting more

TABLE III  
TEMPORAL ACTION LOCALIZATION RESULTS ON THE MSR ACTION DATASET II.

Model	$\alpha = 0.5$
Yu <i>et al.</i> [48]	28.2
Gemert <i>et al.</i> [13]	54.5
Heilbron <i>et al.</i> [5]	60.3
Zhu <i>et al.</i> [55]	61.1
Ours	63.1

accurate temporal boundaries in long temporal range videos. An example of temporal action localization of the MSR II dataset is shown on the top three rows in Figure 10.

### F. Results on the MPII Cooking dataset

On the MPII Cooking dataset, the sliding window method is treated as the baseline method the same as in [55]. The temporal action localization results are listed in Table IV. Our method improves the baseline by 7.8%. Different from the method [28], our method works on the videos with varied temporal length rather than limiting the length of videos to a constant. Especially, for long temporal length videos, our



Fig. 9. The bad prediction results of three action instances (“Billiards”, “Shotput” and “LongJump”) on the THUMOS2014 dataset. We display the ground truth and the prediction results in the gray section.

TABLE IV  
TEMPORAL ACTION LOCALIZATION RESULTS ON THE MPII COOKING DATASET.

Model	$\alpha = 0.5$
Sliding Window	7.9
Gemert <i>et al.</i> [13]	13.1
Richard <i>et al.</i> [28]	14.0
Zhu <i>et al.</i> [55]	14.9
Ours	15.7

method can produce more precise temporal boundaries of action instances. Two examples of the action instances (“Take out from cupboard” and “Cut Slices”) are displayed on the bottom of Figure 10. As shown in Figure 8 and 10, our method can perform well on the complex (THUMOS 2014), simple (MSR ActionII), fine-grained (MSRII Cooking) for temporal action localization datasets.

### G. Impact of Different Components

The effects of different components on the localization results are shown in Table V, Table VI and Table VII. In Table V, “w/o AP-Trees” stands for the method without AP-Trees. In “w/o AP-Trees”, the AP-Trees are replaced with a simple combination of segments, which means the localization results are produced by combining the segments which have the same action label and post-processing with non-maximum suppression. The results using the AP-Trees improve about 2.3%, 3.7%, and 1.4% than without the AP-Trees on the THUMOS2014, MSR ActionII and MPII Cooking datasets, respectively. We also compare our method with TAG [52], and the result also validates the effectiveness of the proposed AP-Trees on predicting more precise temporal boundaries than simple combination of the labeled candidate segments.

In Table VI, the results of different structures of deep neural networks are reported. The networks with different structures

are used to generate label vectors of videos and the AP-Trees are utilized to produce the temporal boundaries of action instances with the label vectors. “C3D” stands for using the C3D network to classify video segments. “IAM” and “Sem” represent using only the spatio-temporal information and using only the high-level semantic features of the segments for classification, respectively. “GMP” is the method where the average pooling layer is replaced with a max pooling layer in creating informative action maps. It is evident that: (1) The spatio-temporal information and the high-level features are complementary. Both the spatio-temporal information and the high-level semantic features can contribute to the labeling of segments in videos; (2) The average pooling performs better than the max pooling in 3D ConvNets for generating informative action maps. In Table VII, “w/o Proposal” and “w/o Loc” stand for the proposed network without the “Proposal” and “Localization” network, respectively. The results show that either “Proposal” or “Localization” network can also improve the precision of action temporal localization.

## VI. CONCLUSION

In this paper, we present a novel framework for temporal action localization in long untrimmed videos based on action pattern trees and deep neural networks. To the best of our knowledge, this is the first work to employ action pattern trees to learn the occurrence frequency and the order of segments in videos and produce the temporal boundaries of action instances. Deep neural networks are trained to capture spatio-temporal information and high-level semantic features to improve the accuracy of labeling segments in videos. The networks are employed to determine whether the segments in a video belong to the action categories or to the background, and to generate the predicted action labels. Experimental results on three benchmarks demonstrate the effectiveness of our method compared to the state-of-the-art, and the influence of the different components of the framework on the localization



TABLE V  
RESULTS ON DIFFERENT TEMPORAL STRUCTURE MODELING METHODS.

	w/o AP-Trees	TAG [52]	Ours
THUMOS 2014	17.8	18.9	20.1
MSR ActionII	59.4	61.6	63.1
MPII Cooking	14.3	14.8	15.7

TABLE VI  
RESULTS ON DIFFERENT STRUCTURES OF DEEP NEURAL NETWORKS.

	C3D	IAM	Sem	GMP	Ours
THUMOS 2014	19.4	19.2	19.3	19.5	20.1
MSR ActionII	61.7	61.1	61.3	62.3	63.1
MPII Cooking	14.8	14.7	14.7	14.7	15.7

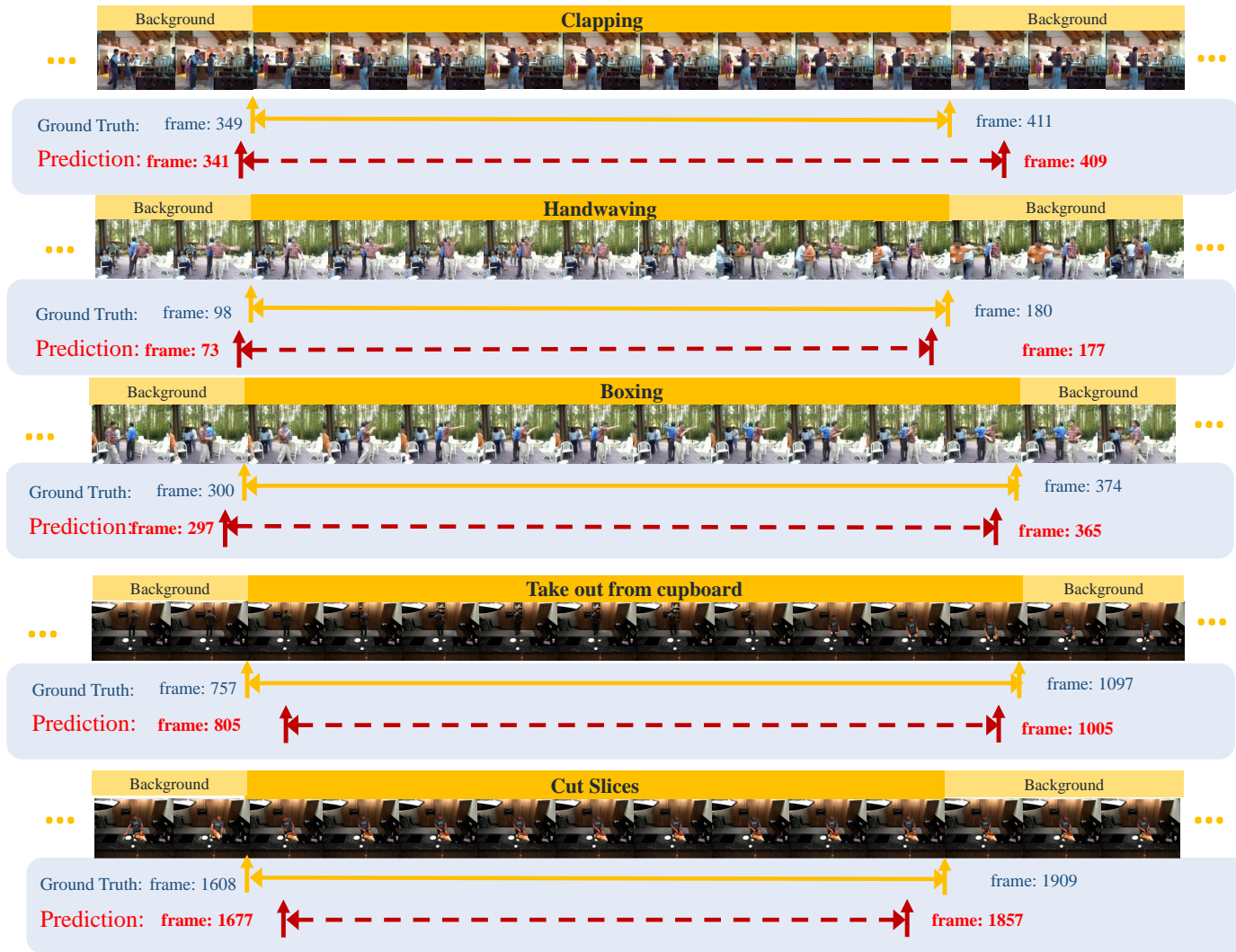


Fig. 10. The prediction results of temporal action localization on the MSR ActionII dataset (top three) and MPII Cooking dataset (bottom two).

TABLE VII  
RESULTS ON DIFFERENT COMPONENTS OF DEEP NEURAL NETWORKS.

	w/o Pro	w/o Loc	Ours
THUMOS 2014	19.2	19.5	20.1
MSR ActionII	62.1	62.3	63.1
MPII Cooking	14.7	14.7	15.7

performance. In future, we plan to apply the AP-Tree model to the spatio-temporal action localization.

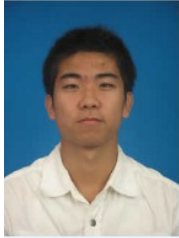
## REFERENCES

- [1] P. ADITYA, "Market basket analysis using fp-growth algorithm in organic medicine store," *Skripsi, Fakultas Ilmu Komputer*, 2016.
- [2] L. Baraldi, C. Grana, and R. Cucchiara, "Recognizing and presenting the storytelling video structure with deep multimodal networks," *IEEE Transactions on Multimedia*, vol. 19, no. 5, pp. 955–968, 2017.
- [3] E. Z. Borzeshi, R. Xu, and M. Piccardi, "Automatic human action recognition in videos by graph embedding," pp. 19–28, 2011.
- [4] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. C. Niebles, "Sst: Single-stream temporal action proposals," in *Computer Vision and Pattern Recognition*, 2017, pp. 6373–6382.
- [5] F. Caba Heilbron, J. Carlos Niebles, and B. Ghanem, "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*, 2016, pp. 1914–1923.
- [6] H.-Y. Chang, J.-C. Lin, M.-L. Cheng, and S.-C. Huang, "A novel incremental data mining algorithm based on fp-growth for big data," in *Proceedings of the International Conference on Networking and Network Applications (NaNA)*. IEEE, 2016, pp. 375–378.
- [7] G. Cheng, Y. Wan, A. N. Saudagar, K. Namuduri, and B. P. Buckles,



- “Advances in human action recognition: A survey,” *Computer Science*, 2015.
- [8] X. Dai, B. Singh, G. Zhang, L. S. Davis, and Y. Q. Chen, “Temporal context network for activity localization in videos,” in *IEEE International Conference on Computer Vision*, 2017, pp. 5727–5736.
- [9] K. Dharmaraajan and M. Dorairangaswamy, “Analysis of fp-growth and apriori algorithms on pattern discovery from weblog data,” in *Proceedings of the IEEE International Conference on Advances in Computer Applications (ICACA)*. IEEE, 2016, pp. 170–174.
- [10] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “Decaf: A deep convolutional activation feature for generic visual recognition,” in *Proceedings of the International Conference on Machine Learning*, 2014, pp. 647–655.
- [11] A. Gaidon, Z. Harchaoui, and C. Schmid, “Temporal localization of actions with actoms,” *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2782–2795, 2013.
- [12] J. Gao, Z. Yang, and R. Nevatia, “Cascaded boundary regression for temporal action detection,” *British Machine Vision Conference*, 2017.
- [13] J. Gemert, M. Jain, E. Gati, C. G. Snoek *et al.*, *Apt: Action localization proposals from dense trajectories*. BMVA Press, 2015.
- [14] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation,” in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 1–12.
- [15] S. Herath, M. Harandi, and F. Porikli, “Going deeper into action recognition: A survey,” *Image and Vision Computing*, vol. 60, 2017.
- [16] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [17] R. Hou, C. Chen, and M. Shah, “Tube convolutional neural network (t-cnn) for action detection in videos,” 2017.
- [18] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, and C. G. Snoek, “Action localization with tubelets from motion,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 740–747.
- [19] S. Ji, W. Xu, M. Yang, and K. Yu, “3d convolutional neural networks for human action recognition,” *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.
- [20] S. Karaman, L. Seidenari, and A. Del Bimbo, “Fast saliency based pooling of fisher encoded dense trajectories,” in *Proceedings of the European Conference on Computer Vision THUMOS Workshop*, vol. 1, 2014, p. 6.
- [21] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.
- [22] T. Lin, X. Zhao, and Z. Shou, “Single shot temporal action detection,” in *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 2017, pp. 988–996.
- [23] J. Liu, A. Shahroudy, D. Xu, and G. Wang, “Spatio-temporal lstm with trust gates for 3d human action recognition,” in *European Conference on Computer Vision*, 2016, pp. 816–833.
- [24] S. Ma, L. Sigal, and S. Sclaroff, “Space-time tree ensemble for action recognition,” in *Computer Vision and Pattern Recognition*, 2015.
- [25] S. Ma, J. Zhang, N. Ikidler-Cinbis, and S. Sclaroff, “Action recognition and localization by hierarchical space-time segments,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2744–2751.
- [26] K. Mikolajczyk and H. Uemura, “Action recognition with appearance-motion features and fast search trees,” *Computer Vision and Image Understanding*, vol. 115, no. 3, pp. 426–438, 2011.
- [27] D. Oneata, J. Verbeek, and C. Schmid, “The lear submission at thumos 2014,” *THUMOS2014 Challenge*, 2014.
- [28] A. Richard and J. Gall, “Temporal action detection using a statistical language model,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3131–3140.
- [29] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele, “A database for fine grained activity detection of cooking activities,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 1194–1201.
- [30] C. Schuldt, I. Laptev, and B. Caputo, “Recognizing human actions: a local svm approach,” in *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 3. IEEE, 2004, pp. 32–36.
- [31] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S. F. Chang, “Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1417–1426.
- [32] Z. Shou, D. Wang, and S.-F. Chang, “Temporal action localization in untrimmed videos via multi-stage cnns,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1049–1058.
- [33] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Advances in Neural Information Processing Systems*, 2014, pp. 568–576.
- [34] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia, “Temporal localization of fine-grained actions in videos by domain transfer from web images,” in *Proceedings of the 23rd ACM International Conference on Multimedia*. ACM, 2015, pp. 371–380.
- [35] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4489–4497.
- [36] T. T. Truyen, D. Q. Phung, S. Venkatesh, and H. H. Bui, “Adaboost.mrf: Boosted markov random forests and application to multilevel activity recognition,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1686–1693.
- [37] B. Wang, D. Chen, B. Shi, J. Zhang, Y. Duan, J. Chen, and R. Hu, “Comprehensive association rules mining of health examination data with an extended fp-growth method,” *Mobile Networks and Applications*, pp. 1–8, 2016.
- [38] H. Wang, X. Wu, and Y. Jia, “Video annotation via image groups from the web,” *IEEE Transactions on Multimedia*, vol. 16, no. 5, pp. 1282–1291, 2014.
- [39] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 3551–3558.
- [40] L. Wang, Y. Qiao, and X. Tang, “Action recognition and detection by combining motion and appearance features,” *THUMOS14 Action Recognition Challenge*, vol. 1, p. 2, 2014.
- [41] X. Wang, L. Gao, P. Wang, X. Sun, and X. Liu, “Two-stream 3d convnet fusion for action recognition in videos with arbitrary size and length,” *IEEE Transactions on Multimedia*, vol. PP, no. 99, pp. 1–1, 2017.
- [42] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine Learning*, vol. 8, no. 3–4, pp. 229–256, 1992.
- [43] H. Xu, A. Das, and K. Saenko, “R-c3d: Region convolutional 3d network for temporal activity detection,” in *The IEEE International Conference on Computer Vision (ICCV)*, vol. 6, 2017, p. 8.
- [44] K. Xu, Y. Lu, H. Zhang, and X. Feng, “Combining nonuniform sampling, hybrid super vector, and random forest with discriminative decision trees for action recognition,” in *IEEE International Conference on Image Processing*, 2015, pp. 3977–3981.
- [45] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei, “End-to-end learning of action detection from frame glimpses in videos,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2678–2687.
- [46] Y. Yi and M. Lin, “Human action recognition with graph-based multiple-instance learning,” *Pattern Recognition*, vol. 53, no. C, pp. 148–162, 2016.
- [47] G. Yu, N. A. Goussies, J. Yuan, and Z. Liu, “Fast action detection via discriminative random forest voting and top-k subvolume search,” *IEEE Transactions on Multimedia*, vol. 13, no. 3, pp. 507–517, 2011.
- [48] G. Yu and J. Yuan, “Fast action proposals for human action detection and search,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1302–1311.
- [49] J. Yuan, Z. Liu, and Y. Wu, “Discriminative video pattern search for efficient action detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, pp. 1728–1743, 2011.
- [50] Z. Yuan, J. C. Stroud, T. Lu, and J. Deng, “Temporal action localization by structured maximal sums,” *arXiv preprint arXiv:1704.04671*, 2017.
- [51] S. Zhang, Y. Yang, J. Xiao, X. Liu, Y. Yang, D. Xie, and Y. Zhuang, “Fusing geometric features for skeleton-based action recognition using multilayer lstm networks,” *IEEE Transactions on Multimedia*, vol. PP, no. 99, pp. 1–1, 2018.
- [52] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin, “Temporal action detection with structured segment networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [53] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2921–2929.
- [54] Z. Zhou, F. Shi, and W. Wu, “Learning spatial and temporal extents of human actions for action detection,” *IEEE Transactions on Multimedia*, vol. 17, no. 4, pp. 512–525, 2015.

- [55] Y. Zhu and S. Newsam, "Efficient action detection in untrimmed videos via multi-task learning," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, 2017, pp. 197–206.



**Hao Song** received the B.S. degree from North China Electric Power University (NCEPU), Baoding, China, in 2012. He is currently pursuing the Ph.D. degree at the Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, under the supervision of Prof. Y. Jia. His research interests include computer vision, machine learning and video retrieval.



**Mei Chen** is an Associate Professor in the Electrical and Computer Engineering Department at the State University of New York, Albany. She was the Intel Principal Investigator for the Intel Science and Technology Center on Embedded Computing hosted at Carnegie Mellon University. Previously she held researcher and research lead positions at Intel Labs, HP Labs, and SRI Sarnoff Corporation. Meis work in computer vision and biomedical image analysis were nominated finalists for 6 Best Paper Awards and won 3. She earned a Ph.D. in Robotics from the School of Computer Science, Carnegie Mellon University, and a M.S. and B.S. from Tsinghua University, Beijing, China.



understanding.

**Xinxiao Wu (M'09)** is an Associate Professor in the School of Computer Science at the Beijing Institute of Technology. She received the B.A. degree in computer science from the Nanjing University of Information Science and Technology in 2005 and the Ph.D. degree in computer science from the Beijing Institute of Technology in 2010. She was a post-doctoral research fellow at Nanyang Technological University, Singapore, from 2010 to 2011. Her current research interests include machine learning, computer vision, and video analysis and

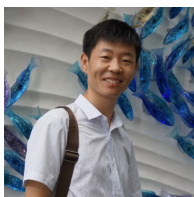


**Bing Zhu** will receive the B.S. degree from Beijing University of Technology (BJUT), Beijing, China, in 2017. He will pursuing the Master degree at the Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, under the supervision of Assoc. Prof. XinXiao Wu. His research interests include computer vision, machine learning and action detection.



His current research interests include computer vision, media computing, and intelligent systems.

**Yunde Jia (M'11)** is Professor of Computer Science at BIT, and serves as the Director of the Beijing Laboratory of Intelligent Information Technology. He received the B.S., M.S., and Ph.D. degrees in Mechatronics from the Beijing Institute of Technology (BIT) in 1983, 1986, and 2000, respectively. He has previously served as the Executive Dean of the School of Computer Science at BIT from 2005 to 2008. He was a Visiting Scientist at Carnegie Mellon University from 1995 to 1997, and a Visiting Fellow at the Australian National University in 2011.



from China Association for Artificial Intelligence (CAAI).

**Yuwei Wu** received the Ph.D. degree in computer science from Beijing Institute of Technology (BIT), Beijing, China, in 2014. He is now an Assistant Professor at School of Computer Science, BIT. From August 2014 to August 2016, he was a post-doctoral research fellow at School of Electrical & Electronic Engineering, Nanyang Technological University (NTU), Singapore. He has strong research interests in computer vision and information retrieval. He received outstanding Ph.D. Thesis award from BIT, and Distinguished Dissertation Award Nominee