Gaussian Temporal Awareness Networks for Action Localization*

Fuchen Long†, Ting Yao‡, Zhaofan Qiu†, Xinmei Tian†, Jiebo Luo§ and Tao Mei‡
†University of Science and Technology of China, Hefei, China
‡JD AI Research, Beijing, China
§University of Rochester, Rochester, NY USA
{longfc.ustc, tingyao.ustc, zhaofanqiu}@gmail.com; xinmei@ustc.edu.cn;
jluo@cs.rochester.edu; tmei@live.com

Abstract

Temporally localizing actions in a video is a fundamental challenge in video understanding. Most existing approaches have often drawn inspiration from image object detection and extended the advances, e.g., SSD and Faster R-CNN, to produce temporal locations of an action in a 1D sequence. Nevertheless, the results can suffer from robustness problem due to the design of predetermined temporal scales, which overlooks the temporal structure of an action and limits the utility on detecting actions with complex variations. In this paper, we propose to address the problem by introducing Gaussian kernels to dynamically optimize temporal scale of each action proposal. Specifically, we present Gaussian Temporal Awareness Networks (GTAN) — a new architecture that novelly integrates the exploitation of temporal structure into an one-stage action localization framework. Technically, GTAN models the temporal structure through learning a set of Gaussian kernels, each for a cell in the feature maps. Each Gaussian kernel corresponds to a particular interval of an action proposal and a mixture of Gaussian kernels could further characterize action proposals with various length. Moreover, the values in each Gaussian curve reflect the contextual contributions to the localization of an action proposal. Extensive experiments are conducted on both THUMOS14 and ActivityNet v1.3 datasets, and superior results are reported when comparing to state-of-the-art approaches. More remarkably, GTAN achieves 1.9% and 1.1% improvements in mAP on testing set of the two datasets.

1. Introduction

With the tremendous increase of online and personal media archives, people are generating, storing and consuming a large collection of videos. The trend encourages the development of effective and efficient algorithms to intelligently parse video data. One fundamental challenge that underlies the success of these advances is action detection in videos from both temporal aspect [6, 9, 17, 30, 39, 43] and spatio-temporal aspect [11, 18]. In this work, the main focus is temporal action detection/localization, which is to locate the exact time stamps of the starting and the ending of an action, and recognize the action with a set of categories.

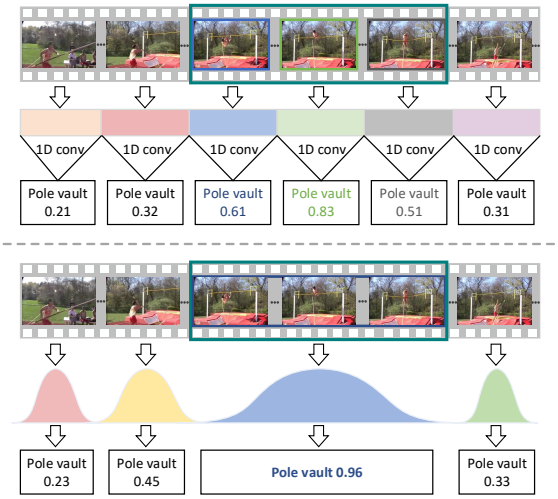


Figure 1. The intuition of a typical one-stage action localization (upper) and our GTAN (lower). The typical method fixes temporal scale in each feature map and seldom explores temporal structure of an action. In contrast, temporal structure is taken into account in our GTAN through learning a set of Gaussian kernels.

One natural way of temporal action localization is to extend image object detection frameworks, e.g., SSD [23] or Faster R-CNN [27], for producing spatial bounding boxes in a 2D image to temporal localization of an action in a 1D sequence [4, 19]. The upper part of Figure 1 conceptualizes a typical process of one-stage action localization. In general, the frame-level or clip-level features in the video sequence are first aggregated into one feature map, and then multiple 1D temporal convolutional layers are devised to increase the

*This work was performed at JD AI Research.

size of temporal receptive fields and predict action proposals. However, the temporal scale corresponding to the cell in each feature map is fixed, making such method unable to capture the inherent temporal structure of an action. As such, one ground-truth action proposal in the green box is detected as three ones in this case. Instead, we propose to alleviate the problem by exploring the temporal structure of an action through learning a Gaussian kernel for each cell, which dynamically indicates a particular interval of an action proposal. A mixture of Gaussian kernels could even be grouped to describe an action, which is more flexible to localize action proposals with various length as illustrated in the bottom part of Figure 1. More importantly, the contextual information is naturally involved with the feature pooling based on the weights in Gaussian curve.

By delving into temporal structure of an action, we present a novel Gaussian Temporal Awareness Networks (GTAN) architecture for one-stage action localization. Given a video, a 3D ConvNet is utilized as the backbone to extract clip-level features, which are sequentially concatenated into a feature map. A couple of convolutional layers plus max-pooling layer are firstly employed to shorten the feature map and increase the temporal size of receptive fields. Then, a cascaded of 1D temporal convolutional layers (anchor layers) continuously shorten the feature map and output anchor feature map, which consists of features of each cell (anchor). On the top of each anchor layer, a Gaussian kernel is learnt for each cell to dynamically predict a particular interval of an action proposal corresponding to that cell. Multiple Gaussian kernels could even be mixed to capture action proposals with arbitrary length. Through Gaussian pooling, the features of each cell is upgraded by aggregating the features of contextual cells weighted by the values in the Gaussian curve for final action proposal prediction. The whole architecture is end-to-end optimized by minimizing one classification loss plus two regression losses, i.e., localization loss and overlap loss.

The main contribution of this work is the design of an one-stage architecture GTAN for addressing the issue of temporal action localization in videos. The solution also leads to the elegant view of how temporal structure of an action should be leveraged for detecting actions with various length and how contextual information should be utilized for boosting temporal localization, which are problems not yet fully understood in the literature.

2. Related Work

We briefly group the related works into two categories: temporal action proposal and temporal action detection. The former focuses on investigating how to precisely localize video segments which contain actions, while the latter further classifies these actions into known classes.

We summarize the approaches on temporal action proposal mainly into two directions: content-independent proposal and content-dependent proposal. The main stream of content-independent proposal algorithms is uniformly or sliding window-ly sampling in a video [24, 33, 41], which leads to huge computations for further classification. In contrast, content-dependent proposal methods, e.g., [3, 5, 7, 8, 21], utilize the label of action proposals during training. For instance, Escorcia *et al.* [5] leverage Long Short-Term Memory cells to learn an appropriate encoding of a video sequence as a set of discriminative states to indicate proposal scores. Though the method avoids running sliding windows of multiple scales, there is still the need of executing an overlapping sliding window that is inapplicable when the video duration is long. To address this problem, Single Stream Temporal proposal (SST) [3] generates proposals with only one single pass by utilizing a recurrent GRU-based model, and Temporal Unit Regression Network (TURN) [8] builds video units in a pyramid manner to avoid window overlapping. Different from the above methods which generate proposals in a fixed multi-scale manner, Boundary Sensitive Network (BSN) [21] localizes the action boundaries based on three actionness curves in a more flexible way. Nevertheless, such actionness-based methods may fail in locating dense and short actions because of the difficulty to discriminate between very close starting and ending peaks in the curve.

Once the localization of action proposals completes, the natural way for temporal action detection is to further classify the proposals into known action classes, making the process in two-stage manner [4, 12, 29, 30, 38, 43]. However, the separate of proposal generation and classification may result in sub-optimal solutions. To further facilitate temporal action detection, there have been several one-stage techniques [2, 19, 40] being proposed recently. For example, Single Stream Temporal Action Detection (SS-TAD) [2] utilizes the Recurrent Neural Network (RNN) based architecture to jointly learn action proposal and classification. Inspired by SSD [23], Lin *et al.* [19] devise 1D temporal convolution to generate multiple temporal action anchors for action proposal and detection. Moreover, with the development of reinforcement learning, Yeung *et al.* [40] explore RNN to learn a glimpse policy for predicting the starting and ending points of actions in an end-to-end manner. Nevertheless, most of one-stage methods are still facing the challenge in localizing all the action proposals due to the predetermined temporal scales.

In short, our approach belongs to one-stage temporal action detection techniques. Different from the aforementioned one-stage methods which often predetermine temporal scales of action proposals, our GTAN in this paper contributes by studying not only learning temporal structure through Gaussian kernels, but also how the contextual information can be better leveraged for action localization.

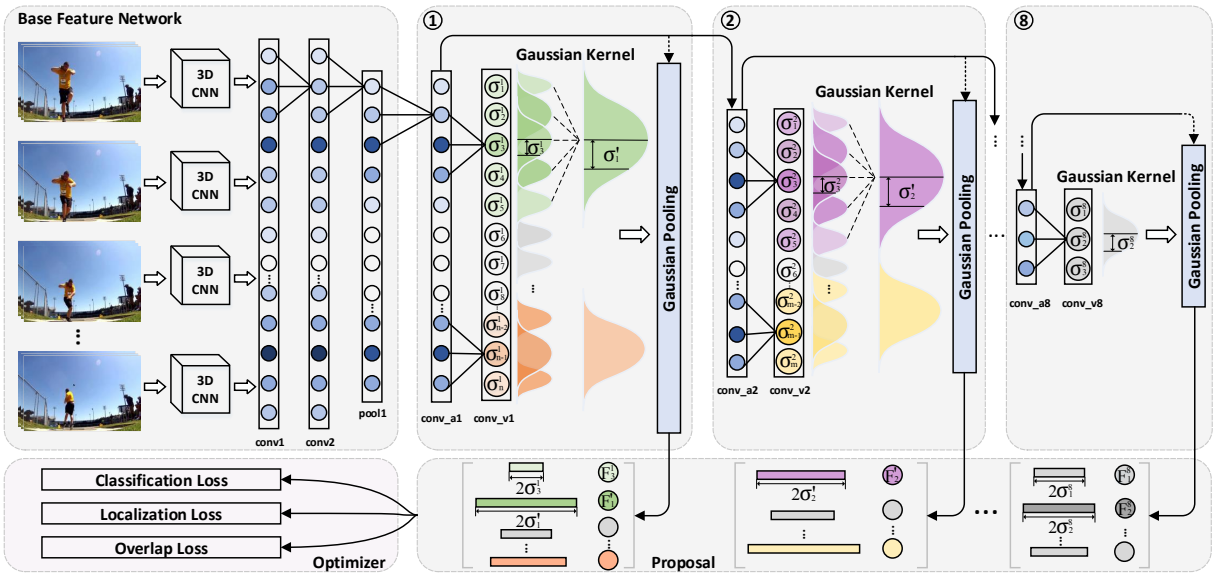


Figure 2. An overview of our Gaussian Temporal Awareness Networks (GTAN) architecture. The input video is encoded into a series of clip-level features via a 3D ConvNet, which are sequentially concatenated as a feature map. Two 1D convolutional layers plus one max-pooling layer are followed to increase the temporal size of receptive fields. Eight 1D convolutional layers are cascaded to generate multiple feature maps in different temporal resolution. On the top of each feature map, a Gaussian kernel is learnt on each cell to predict a particular interval of an action proposal. Moreover, multiple Gaussian kernels with high overlap are mixed to a larger one for detecting long actions with various length. Through Gaussian pooling, the action proposal is generated by aggregating the features of contextual cells weighted by the values in the Gaussian curve. The GTAN is jointly optimized with action classification loss plus two regression losses, i.e., localization loss and overlap loss for each proposal. Better viewed in original color pdf.

3. Gaussian Temporal Awareness Networks

In this section we present the proposed Gaussian Temporal Awareness Networks (GTAN) in detail. Figure 2 illustrates an overview of our architecture for action localization. It consists of two main components: a base feature network and a cascaded of 1D temporal convolutional layers with Gaussian kernels. The base feature network is to extract feature map from sequential video clips, which will be fed into cascaded 1D convolutional layers to generate multiple feature maps in different temporal resolution. For each cell in one feature map, a Gaussian kernel is learnt to control temporal scale of an action proposal corresponding to that cell as training proceeds. Furthermore, a Gaussian Kernel Grouping algorithm is devised to merge multiple Gaussian kernels with high overlap to a larger one for capturing long actions with arbitrary length. Specifically, each action proposal is generated by aggregating the features of contextual cells weighted by the values in the Gaussian curve. The whole network is jointly optimized with action classification loss plus two regression losses, i.e., localization loss and overlap loss, which are utilized to learn action category label, default temporal boundary adjustment and overlap confidence score for each action proposal, respectively.

3.1. Base Feature Network

The ultimate target of action localization is to detect action instances in temporal dimension. Given an input video, we first extract clip-level features from continuous clips via a 3D ConvNet which could capture both appearance and motion information of the video. Specifically, a sequence of features $\{f_i\}_{i=0}^{T-1}$ are extracted from 3D ConvNet, where T is the temporal length. We concatenate all the features into one feature map and then feed the map into two 1D convolutional layers ("conv1" and "conv2" with temporal kernel size 3, stride 1) plus one max-pooling layer ("pool1" with temporal kernel size 3, stride 2) to increase the temporal size of receptive fields. The base feature network is composed of 3D ConvNet, two 1D convolutional layers and max-pooling layer. The outputs of the base feature network are further exploited for action proposal generation.

3.2. Gaussian Kernel Learning

Given the feature map output from the base feature network, a natural way for one-stage action localization is to stack 1D temporal convolutional layers (anchor layers) to generate proposals (anchors) for classification and boundary regression. This kind of structure with predetermined temporal scale in each anchor layer can capture action proposals whose temporal intervals are well aligned with the size of receptive fields, however, posts difficulty to the detection of proposals with various length. The design limits the utility on localizing actions with complex variations.

To address this issue, we introduce temporal Gaussian kernel to dynamically control the temporal scales of proposals in each feature map. In particular, as shown in Figure

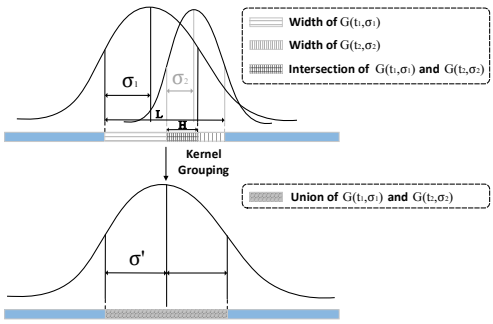


Figure 3. Visualization of Gaussian Kernel Grouping.

2, eight 1D temporal convolutional layers (anchor layers) are first cascaded for action proposal generation in different temporal resolution. For each cell in the feature map of the anchor layer, a Gaussian kernel is learnt to predict a particular interval of an action proposal corresponding to that cell. Formally, we denote the feature map of j-th convolutional layer as $\{f_i\}_{i=0}^{T^j-1} \in \mathbb{R}^{T^j \times D^j}$, $1 \leq j \leq 8$, where T^j and D^j are the temporal length and feature dimension of the feature map. For a proposal P_t^j whose center location is t, we leverage its temporal scale by a Gaussian kernel G_t^j. The standard deviation σ_t^j of G_t^j is learnt via a 1D convolutional layer on a $3 \times D^j$ feature map cell, and the value is constrained within the range $(0, 1)$ through a sigmoid operation. The weights of the Gaussian kernel G_t^j are defined as

$$\begin{aligned} & W_t^j[i] = \frac{1}{Z} \exp(-\frac{(p_i - \mu_t)^2}{2{\sigma_t^j}^2}) \,, \\ & s.t. \quad p_i = \frac{i}{T^j}, \; \mu_t = \frac{t}{T^j} \,, \\ & \qquad i \in \{0, 1, ..., T^j - 1\}, \quad t \in \{0, 1, ..., T^j - 1\}, \end{aligned} \tag{1}$$

where Z is the normalizing constant. Taking the spirit from the theory that the σ_t^j could be considered as a measure of width (Root Mean Square width, RMS) in Gaussian kernel G_t^j, we utilize σ_t^j as the interval measure of action proposal P_t^j. Specifically, the σ_t^j can be multiplied with a certain ratio to represent the default temporal boundary:

$$a_c = (t + 0.5)/T^j, \quad a_w = r_d \cdot 2\sigma_t^j / T^j, \tag{2}$$

where a_c and a_w are the center location and width of default temporal boundary and r_d represents temporal scale ratio. The W_t^j is also utilized for feature aggregation with a pooling mechanism to generate action proposals, which will be elaborated in Section 3.4.

Compared to the conventional 1D convolutional anchor layer which fixes the temporal scale as $1/T^j$ in j-th layer, ours employs the dynamic temporal scales by leveraging the learned Gaussian kernel of each proposal to explore the action instances with complex variations.

3.3. Gaussian Kernel Grouping

Through learning temporal Gaussian kernels, the temporal scales of most action instances can be characterized with the predicted standard deviation. However, if the learned Gaussian kernels span and overlap with each other, that may implicitly indicate a long action centered at a flexible position among these Gaussian kernels. In other words, utilizing the center locations of these original Gaussian kernels to represent this long proposal may not be appropriate. To alleviate this issue, we attempt to generate a set of new Gaussian kernels to predict center location and temporal scales of proposals for long action. Inspired by the idea of temporal actionness grouping in [43], we propose a novel Gaussian Kernel Grouping algorithm for this target.

Algorithm 1 Gaussian Kernel Grouping

Input:
Original Gaussian kernel set $\mathbb{S} = \{G(t_i, \sigma_i)\}_{i=0}^{T-1}$;
Intersection over Union (IoU) threshold ε;
Output:
Mixed Gaussian kernel set $\mathbb{G}$;
Choose the beginning grouping position $p = 0$;
Initialize mixed Gaussian kernel set $\mathbb{G} = \varnothing$;
Initialize base Gaussian kernel $G_{bs} = G(t_p, \sigma_p)$, the ending grouping position $z = p + 1$;
while $p \leq T - 1$ **do**
  Compute IoU value O between kernel G_{bs} and $G(t_z, \sigma_z)$;
  if $O > \varepsilon$ **then**
    Group G_{bs} and $G(t_z, \sigma_z)$ to G' according to Eq.(3), replace G_{bs} with the new mixed kernel G';
  else
    Add kernel G_{bs} to mixed kernel set $\mathbb{G}$;
    $p = z$, $G_{bs} = G(t_p, \sigma_p)$;
  end if
  $z = z + 1$;
end while
return $\mathbb{G}$

Figure 3 illustrates the process of temporal Gaussian Kernel Grouping. Given two adjacent Gaussian kernels $G(t_1, \sigma_1)$ and $G(t_2, \sigma_2)$ whose center location and standard deviation are t and σ, we compute the temporal intersection and union between two kernels by using the width a_w of the default temporal boundary defined in Section 3.2. In upper part of Figure 3, the length of temporal intersection between two kernels is H, while the length of union is L. If the Intersection over Union (IoU) between the two kernels H/L exceeds a certain threshold ε, we merge them into one Gaussian kernel (bottom part of Figure 3). The new mixed Gaussian kernel is formulated as follows

$$\begin{aligned} & W[i] = \frac{1}{Z} \exp(-\frac{(p_i - \mu')^2}{2{\sigma'}^2}) \,, \\ & s.t. \quad p_i = \frac{i}{T}, \quad \mu' = \frac{t_1 + t_2}{2 \cdot T} \,, \quad \sigma' = \frac{L}{2} \,, \\ & \qquad i \in \{0, 1, ..., T - 1\}. \end{aligned} \tag{3}$$

In each feature map, Algorithm 1 details the grouping steps to generate merged kernels.

3.4. Gaussian Pooling

With the learned and mixed Gaussian kernels, we calculate the weighted sum of the feature map based on the values in Gaussian curve and obtain the aggregated feature F. Specifically, given the weighting coefficients W_t^j of

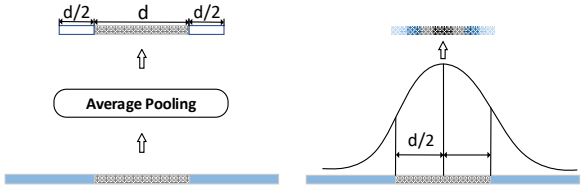


Figure 4. Comparisons of manual extension plus average-pooling strategy (left) and Gaussian pooling strategy (right) for involving temporal contextual information of action proposals.

Gaussian kernel G_t^j at center location t in j-th layer, the aggregated feature for proposal P_t^j is formulated as

$$F_t^j = \frac{1}{T^j} \sum\nolimits_{i=0}^{T^j-1} W_t^j[i] \cdot f_i, \tag{4}$$

where the representation F_t^j is further exploited for the action classification and temporal boundary regression.

The above Gaussian pooling mechanism inherently takes the contextual contributions around each action proposal into account. In contrast to the manual extension plus average-pooling strategy to capture video context information (left part of Figure 4), ours provides an elegant alternative to adaptively learn the weighted representation (right part of Figure 4) based on the importance.

3.5. Network Optimization

Given the representation of each proposal from Gaussian pooling, three 1D convolutional layers are utilized in parallel to predict action classification scores, localization parameters and overlap parameter, respectively. Action classification scores $\mathbf{y^a} = [y_0^a, y_1^a, ..., y_C^a]$ indicate the probabilities belonging to C action classes plus one "background" class. Localization parameters $(\Delta c, \Delta w)$ denote temporal offsets relative to default center location a_c and width a_w, which are leveraged to adjust the temporal coordinate

$$\varphi_c = a_c + \alpha_1 a_w \Delta c \quad \text{and} \quad \varphi_w = a_w \exp\left(\alpha_2 \Delta w\right), \tag{5}$$

where φ_c, φ_w are refined center location and width of the proposal. The α_1, α_2 are utilized to control the impact of temporal offsets. In particular, we define an overlap parameter y_{ov} to represent the precise IoU prediction of the proposal, which benefits the proposal re-ranking in prediction.

In the training stage, we accumulate all the proposals from Gaussian pooling and produce the action instances through prediction layer. The overall training objective in our GTAN is formulated as a multi-task loss by integrating action classification loss (L_{cls}) and two regression losses, i.e., localization loss (L_{loc}) and overlap loss (L_{ov}):

$$L = L_{cls} + \beta L_{loc} + \gamma L_{ov}, \tag{6}$$

where β and γ are the trade-off parameters. Specifically, we measure the classification loss L_{cls} via the softmax loss:

$$L_{cls} = -\sum_{n=0}^{C} I_{n=c} \log(y_n^a), \tag{7}$$

where indicator function $I_{n=c} = 1$ if n equals to ground truth action label c, otherwise $I_{n=c} = 0$. We denote g_{iou} as the IoU between default temporal boundary of this proposal and its corresponding closest ground truth. If the g_{iou} of this proposal is larger than 0.8, we set it as a foreground sample. If g_{iou} is lower than 0.3, it will be set as background sample. The ratio between foreground and background samples is set as 1.0 during training. The localization loss is devised as Smooth L1 loss [10] (S_{L1}) between the predicted foreground proposal and the closest ground truth instance of the proposal, which is computed by

$$L_{loc} = S_{L1}(\varphi_c - g_c) + S_{L1}(\varphi_w - g_w), \tag{8}$$

where g_c and g_w represents the center location and width of the proposal's closest ground truth instance, respectively. For overlap loss, we adopt the mean square error (MSE) loss to optimize it as follows:

$$L_{ov} = (y_{ov} - g_{iou})^2. \tag{9}$$

Eventually, the whole network is trained in an end-to-end manner by penalizing the three losses.

3.6. Prediction and Post-processing

During prediction of action localization, the final ranking score y_f of each candidate action proposal depends on both action classification scores $\mathbf{y^a}$ and overlap parameter y_{ov}:

$$y_f = \max(\mathbf{y^a}) \cdot y_{ov}. \tag{10}$$

Given the predicted action instance $\phi = \{\varphi_c, \varphi_w, C_a, y_f\}$ with refined boundary (φ_c, φ_w), predicted action label C_a, and ranking score y_f, we employ the soft non-maximum suppression (soft-NMS) [1] for post-processing. In each iteration of soft-NMS, we represent the action instance with the maximum ranking score y_{f_m} as ϕ_m. The ranking score y_{f_k} of other instance ϕ_k will be decreased or not, according to the IoU computed with ϕ_m:

$$y'_{f_k} = \begin{cases} y_{f_k} & , \text{ if } iou(\phi_k, \phi_m) < \rho \\ y_{f_k} \cdot e^{-\frac{iou(\phi_k, \phi_m)^2}{\xi}} & , \text{ if } iou(\phi_k, \phi_m) \geq \rho \end{cases}, \tag{11}$$

where ξ is the decay parameter and ρ is the NMS threshold.

4. Experiments

We empirically verify the merit of our GTAN by conducting the experiments of temporal action localization on two popular video recognition benchmarks, i.e., ActivityNet v1.3 [13] and THUMOS14 [16].

4.1. Datasets

The **ActivityNet v1.3** dataset contains 19,994 videos in 200 classes collected from YouTube. The dataset is divided into three disjoint subsets: training, validation and testing,

Table 1. The details of 1D temporal convolutional (anchor) layers. RF represents the size of receptive fields.

id	type	kernel size	#channels	#stride	RF
1	conv_a1	3	512	2	11
2	conv_a2	3	512	2	19
3	conv_a3	3	1024	2	35
4	conv_a4	3	1024	2	67
5	conv_a5	3	2048	2	131
6	conv_a6	3	2048	2	259
7	conv_a7	3	4096	2	515
8	conv_a8	3	4096	2	1027

by 2:1:1. All the videos in the dataset have temporal annotations. The labels of testing set are not publicly available and the performances of action localization on ActivityNet dataset are reported on validation set. The **THUMOS14** dataset has 1,010 videos for validation and 1,574 videos for testing from 20 classes. Among all the videos, there are 220 and 212 videos with temporal annotations in validation and testing set, respectively. Following [43], we train the model on validation set and perform evaluation on testing set.

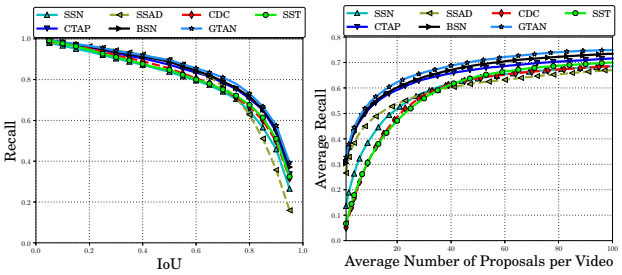
4.2. Experimental Settings

Implementations. We utilize Pseudo-3D [26] network as our 3D backbone. The network input is a 16-frame clip and the sample rate of frames is set as 8. The 2,048-way outputs from pool5 layer are extracted as clip-level features. Table 1 summarizes the structures of 1D anchor layers. Moreover, we choose three temporal scale ratios $\{r_d\}_{d=1}^{3} = [2^0, 2^{1/3}, 2^{2/3}]$ derived from [22]. The IoU threshold ε in Gaussian grouping is set as 0.7 by cross validation. The balancing parameters β and γ are also determined on a validation set and set as 2.0 and 75. ξ and ρ are set as 0.8 and 0.75 in soft-NMS. The parameter α_1 and α_2 are all set as 1.0 by cross validation. We implement GTAN on Caffe [15] platform. In all the experiments, our networks are trained by utilizing stochastic gradient descent (SGD) with 0.9 momentum. The initial learning rate is set as 0.001, and decreased by 10% after every $2.5k$ iterations on THUMOS14 and $10k$ iterations on ActivityNet. The mini-batch size is 16 and the weight decay parameter is 0.0001.

Evaluation Metrics. We follow the official evaluation metrics in each dataset for action detection task. On ActivityNet v1.3, the mean average precision (mAP) values with IoU thresholds between 0.5 and 0.95 (inclusive) with a step size 0.05 are exploited for comparison. On THUMOS14, the mAP with IoU threshold 0.5 is measured. We evaluate performances on top-100 and top-200 returned proposals in ActivityNet v1.3 and THUMOS14, respectively.

4.3. Evaluation on Temporal Action Proposal

We first examine the performances on temporal action proposal task, which is to only assess the boundary quality of action proposals, regardless of action classes. We compare the following advanced approaches: (1) Structure Segment Network (SSN) [43] generates action proposals by temporal actionness grouping. (2) Single Shot Action Detection (SSAD) [19] is the 1D variant version of Single Shot Detection [23], which generates action proposals by multiple temporal anchor layers. (3) Convolution-De-Convolution Network (CDC) [29] builds a 3D Conv-Deconv network to precisely localize the boundary of action instances at frame level. (4) Boundary Sensitive Network (BSN) [21] locates temporal boundaries with three actionness curves and reranks proposals with neural networks. (5) Single Stream Temporal action proposal (SST) [3] builds a RNN-based action proposal network, which could be implemented in a single stream over long video sequences to produce action proposals. (6) Complementary Temporal Action Proposal (CTAP) [7] balances the advantages and disadvantages between sliding window and actionness grouping approaches for final action proposal.



(a) (b)

Figure 5. (a) Recall-IoU and (b) AR-AN curve on ActivityNet.

Table 2. AR and AUC values on action proposal. IoU threshold: [0.5:0.05:1.0] for THUMOS14, [0.5:0.05:0.95] for ActivityNet.

Approach	THUMOS14	ActivityNet		ActivityNet (test server)
	AR	AR	AUC	AUC
SST [3]	37.9	-	-	-
CTAP [7]	50.1	73.2	65.7	-
BSN [21]	53.2	74.2	66.2	66.3
GTAN	**54.3**	**74.8**	**67.1**	**67.4**

We adopt the standard metric of Average Recall in different IoU (AR) for action proposal on both datasets. Moreover, following the official evaluations in ActivityNet, we plot both Recall-IoU curve and Average Recall vs. Average Number of proposals per video (AR-AN) curve in Figure 5. In addition to AR metric, the area under AR-AN curve (AUC) is also reported in Table 2 as AUC is the measure on test server of ActivityNet. Overall, the performances across different metrics and two datasets consistently indicate that our GTAN leads to performance boost against baselines. In particular, AR of GTAN achieves 54.3% and 74.8% on THUMOS14 and ActivityNet respectively, making the absolute improvement over the best competitor BSN by 1.1% and 0.6%. GTAN surpasses BSN by 1.1% in AUC when evaluating on online ActivityNet test server. The results demonstrate the advantages of exploiting temporal structure for localizing actions. Furthermore, as shown in Figure 5, the improvements are constantly attained across different IoU. In terms of AR-AN curve, GTAN also exhibits better

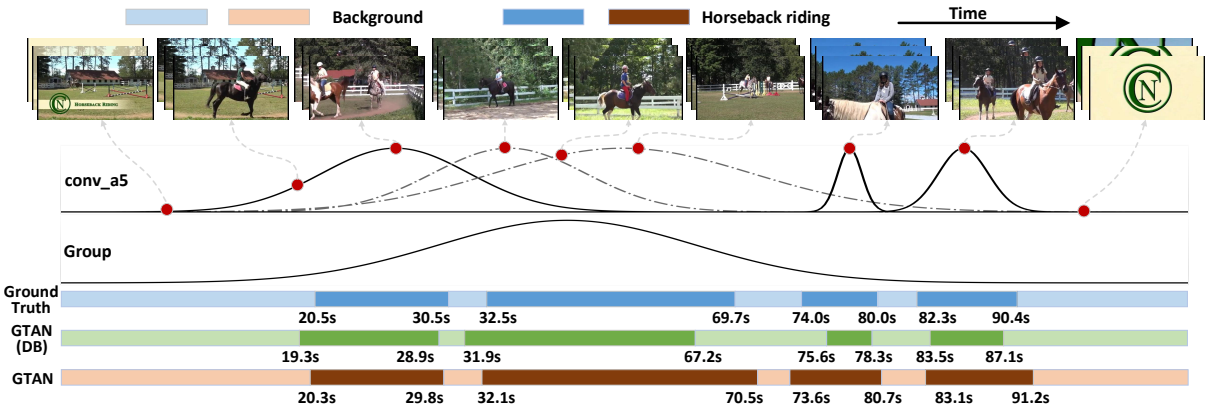


Figure 6. Visualization of action localization on a video example from ActivityNet by GTAN. The Gaussian kernels are learnt on the outputs of "conv_a5" layer. The second and third kernels are mixed into a larger one. The default boxes (DB) are predicted by Gaussian kernels.

Table 3. Performance contribution of each design in GTAN.

Approach	THUMOS14			ActivityNet v1.3		
Fixed Scale	✓			✓		
Gaussian Kernel		✓	✓		✓	✓
Gaussian Grouping			✓			✓
mAP	33.5	37.1	38.2	29.8	31.6	34.3

Table 4. The evaluations of Gaussian grouping on actions with different lengths. GTAN$^-$ excludes Gaussian grouping in GTAN.

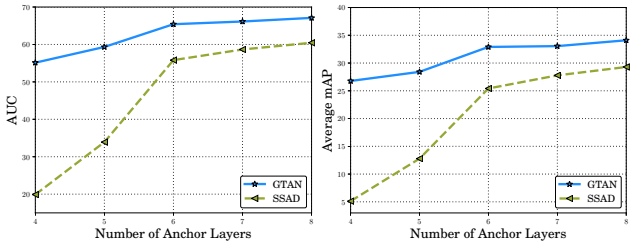
Approach	THUMOS14		ActivityNet v1.3	
	≥ 128	All	≥ 2048	All
GTAN$^-$	22.1	37.1	49.4	31.6
GTAN	25.9	38.2	54.2	34.3

performance on different number of top returned proposals. Even in the case when only less than 10 proposals are returned, GTAN still shows apparent improvements, indicating that GTAN is benefited from the mechanism of dynamically optimizing temporal scale of each proposal and the correct proposals are ranked at the top.

4.4. Evaluation on Gaussian Kernel and Grouping

Next, we study how each design in GTAN influences the overall performance on temporal action localization task. Fixed Scale simply employs a fixed temporal interval for each cell or anchor in an anchor layer and such way is adopted in SSAD. Gaussian Kernel leverages the idea of learning one Gaussian kernel for each anchor to model temporal structure of an action and dynamically predict temporal scale of each action proposal. Gaussian Grouping further mixes multiple Gaussian kernels to characterize action proposals with various length. In the latter two cases, Gaussian pooling is utilized to augment the features of each anchor with contextual information.

Table 3 details the mAP performances by considering one more factor in GTAN on both datasets. Gaussian Kernel successfully boosts up the mAP performance from 33.5% to 37.1% and from 29.8% to 31.6% on THUMOS14 and ActivityNet v1.3, respectively. This somewhat reveals the weakness of Fixed Scale, where the temporal scale of each anchor is independent of temporal property of the action proposal. Gaussian Kernel, in comparison, models temporal structure and predicts a particular interval for each anchor on the fly. As such, the temporal localization or boundary of each action proposal is more accurate. Moreover, the features of each action proposal are simultaneously enhanced by contextual aggregation through Gaussian pooling and lead to better action classification. Gaussian grouping further contributes a mAP increase of 1.1% and 2.7%, respectively. The results verify the effectiveness and flexibility of mixing multiple Gaussian kernels to capture action proposals with arbitrary length. To better validate the impact of Gaussian grouping, we additionally evaluate GTAN on long action proposals. Here, we consider actions longer than 128 frames in THUMOS14 and 2048 frames in ActivityNet v1.3 as long actions, since the average duration of action instances in THUMOS14 is ∼4 seconds which is much smaller than that (∼50 seconds) of ActivityNet. Table 4 shows the mAP comparisons between GTAN and its variant GTAN$^-$ which excludes Gaussian grouping. As expected, larger degree of improvement is attained on long action proposals by involving Gaussian grouping.



(a) (b)

Figure 7. (a) AUC and (b) Average mAP performances of SSAD and GTAN with different number of anchor layers on temporal action proposal and localization tasks in ActivityNet.

4.5. Evaluation on the Number of Anchor Layers

In existing one-stage methods, e.g., SSAD, temporal scale is fixed in each anchor layer and the expansion of multiple temporal scales is implemented through increasing the number of anchor layers. Instead, our GTAN learns

Table 5. Performance comparisons of temporal action detection on THUMOS14, measured by mAP at different IoU thresholds α.

THUMOS14, mAP@α					
Approach	0.1	0.2	0.3	0.4	0.5
Two-stage Action Localization					
Wang *et.al.* [35]	18.2	17.0	14.0	11.7	8.3
FTP [14]	-	-	-	-	13.5
DAP [5]	-	-	-	-	13.9
Oneata *et.al.* [25]	36.6	33.6	27.0	20.8	14.4
Yuan *et.al.* [41]	51.4	42.6	33.6	26.1	18.8
S-CNN [30]	47.7	43.5	36.3	28.7	19.0
SST [3]	-	-	37.8	-	23.0
CDC [29]	-	-	40.1	29.4	23.3
TURN [8]	54.0	50.9	44.1	34.9	25.6
R-C3D [38]	54.5	51.5	44.8	35.6	28.9
SSN [43]	66.0	59.4	51.9	41.0	29.8
CTAP [7]	-	-	-	-	29.9
BSN [21]	-	-	53.5	45.0	36.9
One-stage Action Localization					
Richard *et.al.* [28]	39.7	35.7	30.0	23.2	15.2
Yeung *et.al.* [40]	48.9	44.0	36.0	26.4	17.1
SMS [42]	51.0	45.2	36.5	27.8	17.8
SSAD [19]	50.1	47.8	43.0	35.0	24.6
SS-TAD [2]	-	-	45.7	-	29.2
GTAN (C3D)	67.2	61.1	56.9	46.5	37.9
GTAN	**69.1**	**63.7**	**57.8**	**47.2**	**38.8**

one Gaussian kernel for each anchor in every anchor layer and dynamically predicts temporal scale of the action proposal corresponding to each anchor. The grouping of multiple Gaussian kernels makes the temporal scale more flexible. Even with a small number of anchor layers, our GTAN should be more responsible to localize action proposals with various length in theory. Figure 7 empirically compares the performances between SSAD and our GTAN on ActivityNet v1.3 when capitalizing on different number of anchor layers. As indicated by the results, GTAN consistently outperforms SSAD across different depths of anchor layers from 4 to 8 on both temporal action proposal and localization tasks. In general, more anchor layers provide better AUC and mAP performances. It is expected that the performance of SSAD decreases more sharply than that of GTAN when reducing the number of anchor layers. In the extreme case of 4 layers, GTAN still achieves 26.77% in average mAP while SSAD only reaches 5.12%, which again confirms the advantage of exploring temporal structure and predicting temporal scale of action proposals.

4.6. Comparisons with State-of-the-Art

We compare with several state-of-the-art techniques on THUMOS14 and ActivityNet v1.3 datasets. Table 5 lists the mAP performances with different IoU thresholds on THUMOS14. For fair comparison, we additionally implement GTAN using C3D [34] as 3D ConvNet backbone. The results across different IoU values consistently indicate that GTAN exhibits better performance than others. In particular, the mAP@0.5 of GTAN achieve 37.9% with C3D backbone, making the improvements over one-stage approaches SSAD and SS-TAD by 13.3% and 8.7%, which also employ C3D. Compared to the most advanced two-stage method BSN, our GTAN leads to 1.0% and 1.9% performance gains with C3D and P3D backbone, respectively. The superior results of GTAN demonstrate the advantages of modeling temporal structure of actions through Gaussian kernel.

Table 6. Comparisons of temporal action detection on ActivityNet.

ActivityNet v1.3, mAP					
Approach	validation				testing
	0.5	0.75	0.95	Average	Average
Wang *et.al.* [36]	45.11	4.11	0.05	16.41	14.62
Singh *et.al.* [31]	26.01	15.22	2.61	14.62	17.68
Singh *et.al.* [32]	22.71	10.82	0.33	11.31	17.83
CDC [29]	45.30	26.00	0.20	23.80	22.90
TAG-D [37]	39.12	23.48	5.49	23.98	26.05
SSN [43]	-	-	-	-	28.28
Lin *et.al.* [20]	48.99	32.91	7.87	32.26	33.40
BSN [21]	52.50	33.53	8.85	33.72	34.42
GTAN	**52.61**	**34.14**	**8.91**	**34.31**	**35.54**

On ActivityNet v1.3, we summarize the performance comparisons on both validation and testing set in Table 6. For testing set, we submitted the results of GTAN to online ActivityNet test server and evaluated the performance on the localization task. Similarly, GTAN surpasses the best competitor BSN by 0.6% and 1.1% on validation and testing set, respectively. Moreover, our one-stage GTAN is potentially simpler and faster than two-stage solutions, and tends to be more applicable to action localization in videos.

Figure 6 showcases temporal localization results of one video from ActivityNet. The Gaussian kernels and grouping learnt on the outputs of "conv_a5" layer are also visualized. As shown in the Figure, Gaussian kernels nicely capture the temporal structure of each action proposal and predict accurate default boxes for the final regression and classification.

5. Conclusions

We have presented Gaussian Temporal Awareness Networks (GTAN) which aim to explore temporal structure of actions for temporal action localization. Particularly, we study the problem of modeling temporal structure through learning a set of Gaussian kernels to dynamically predict temporal scale of each action proposal. To verify our claim, we have devised an one-stage action localization framework which measures one Gaussian kernel for each cell in every anchor layer. Multiple Gaussian kernels could be even mixed for the purpose of representing action proposals with various length. Another advantage of using Gaussian kernel is to enhance features of action proposals by leveraging contextual information through Gaussian pooling, which benefits the final regression and classification. Experiments conducted on two video datasets, i.e., THUMOS14 and ActivityNet v1.3, validate our proposal and analysis. Performance improvements are also observed when comparing to both one-stage and two-stage advanced techniques.

Acknowledgments This work was supported in part by the National Key R&D Program of China under contract No. 2017YFB1002203 and NSFC No. 61872329.

References

[1] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-NMS – Improving Object Detection With One Line of Code. In *ICCV*, 2017.

[2] Shyamal Buch, Victor Escorcia, Bernard Ghanem, Li Fei-Fei, and Juan Carlos Niebles. End-to-End, Single-Stream Temporal Action Detection in Untrimmed Videos. In *BMVC*, 2017.

[3] Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. SST: Single-Stream Temporal Action Proposals. In *CVPR*, 2017.

[4] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A. Ross, Jia Deng, and Rahul Sukthankar. Rethinking the Faster R-CNN Architecture for Temporal Action Localization. In *CVPR*, 2018.

[5] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. DAPs: Deep Action Proposals for Action Understanding. In *ECCV*, 2016.

[6] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Temporal Localization of Actions with Actoms. *IEEE Trans. on PAMI*, 35(11):2782–2795, 2013.

[7] Jiyang Gao, Kan Chen, and Ram Nevatia. CFAP: Complementary Temporal Action Proposal Generation. In *ECCV*, 2018.

[8] Jiyang Gao, Zhenheng Yang, Chen Sun, Kan Chen, and Ram Nevatia. TURN TAP: Temporal Unit Regression Network for Temporal Action Proposals. In *ICCV*, 2017.

[9] Roeland De Geest, Efstratios Gavves, Amir Ghodrati, Zhenyang Li, Cees Snoek, and Tinne Tuytelaars. Online Action Detection. In *ECCV*, 2016.

[10] Ross Girshick. Fast R-CNN. In *ICCV*, 2015.

[11] Georgia Gkioxari and Jitendra Malik. Finding Action Tubes. In *CVPR*, 2015.

[12] Fabian Caba Heilbron, Wayner Barrios, Victor Escorica, and Bernard Ghanem. SCC: Semantic Context Cascade for Efficient Action Detection. In *CVPR*, 2017.

[13] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding. In *CVPR*, 2015.

[14] Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Fast Temporal Activity Proposals for Efficient Detection of Human Actions in Untrimmed Videos. In *CVPR*, 2016.

[15] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[16] Yu-Gang Jiang, Jingen Liu, Amir R.Zamir, and George Toderici. THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/THUMOS14`, 2014.

[17] Colin Lea, Rene Vidal Michael D. Flynn, Austin Reiter, and Gregory D. Hager. Temporal Convolutional Netowrk for Action Segmentation and Detection. In *CVPR*, 2017.

[18] Dong Li, Zhaofan Qiu, Qi Dai, Ting Yao, and Tao Mei. Recurrent Tubelet Proposal and Recognition Networks for Action Detection. In *ECCV*, 2018.

[19] Tianwei Lin, Xu Zhao, and Zheng Shou. Single Shot Temporal Action Detection. In *ACM MM*, 2017.

[20] Tianwei Lin, Xu Zhao, and Zheng Shou. Temporal convolution based action proposal: Submission to activitynet 2017. *arXiv preprint arXiv:1707.06750*, 2017.

[21] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. BSN: Boundary Sensitive Network for Temporal Action Proposal Generation. In *ECCV*, 2018.

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal Loss for Dense Object Detection. In *ICCV*, 2017.

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In *ECCV*, 2016.

[24] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. Action and Event Recognition with Fisher Vectors on a Compact Feature Set. In *ICCV*, 2013.

[25] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. The LEAR submission at Thumos 2014. In *ECCV THUMOS Challenge Workshop*, 2014.

[26] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks. In *ICCV*, 2017.

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*, 2015.

[28] Alexander Richard and Juergen Gall. Temporal Action Detection using a Statistical Language Model. In *CVPR*, 2016.

[29] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. CDC: Convolutional-De-Convolutional Network for Precise Temporal Action Localization in Untrimmed Videos. In *CVPR*, 2017.

[30] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal Action Localization in Untrimmed Videos via Multi-stage CNNs. In *CVPR*, 2016.

[31] Bharat Singh, Tim K. Marks, Michael Jones, Oncel Tuzel, and Ming Shao. A Multi-Stream Bi-Directional Recurrent Neural Network for Fine-Grained Action Detection. In *CVPR*, 2016.

[32] Gurkirt Singh and Fabio Cuzzolin. Untrimmed Video Classification for Activity Detection: submission to ActivityNet Challenge. *arXiv preprint arXiv:1607.01979*, 2016.

[33] Kevin Tang, Bangpeng Yao, Li Fei-Fei, and Daphne Koller. Combining the Right Features for Complex Event Recognition. In *ICCV*, 2013.

[34] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In *ICCV*, 2015.

[35] Liming Wang, Yu Qiao, and Xiaoou Tang. Action Recognition and Detection by Combining Motion and Apperance Feature. In *ECCV THUMOS Challenge Workshop*, 2014.

[36] Ruxing Wang and Dacheng Tao. UTS at activitynet 2016. In *CVPR ActivityNet Challenge Workshop*, 2016.

[37] Yuanjun Xiong, Yue Zhao, Limin Wang, Dahua Lin, and Xiaoou Tang. A Pursuit of Temporal Accuracy in General Activity Detection. *arXiv preprint arXiv:1703.02716*, 2017.

[38] Huijuan Xu, Abir Das, and Kate Saenko. R-C3D: Region Convolutional 3D Network for Temporal Activity Detection. In *ICCV*, 2017.

[39] Ting Yao, Yehao Li, Zhaofan Qiu, Fuchen Long, Yingwei Pan, Dong Li, and Tao Mei. MSR Asia MSM at ActivityNet Challenge 2017: Trimmed Action Recognition, Temporal Action Proposals and Dense-Captioning Events in Videos. In *CVPR ActivityNet Challenge Workshop*, 2017.

[40] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end Learning of Action Detection from Frame Glimpses in Videos. In *CVPR*, 2016.

[41] Jun Yuan, Bingbing Ni, Xiaokang Yang, and Ashraf A.Kassim. Temporal Action Localization With Pyramid of Score Distribution Features. In *CVPR*, 2016.

[42] Zehuan Yuan, Jonathan C. Stroud, Tong Lu, and Jia Deng. Temporal Action Localization by Structured Maximal Sums. In *CVPR*, 2017.

[43] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal Action Detection with Structured Segment Networks. In *ICCV*, 2017.