Temporal Action Detection with Structured Segment Networks

Yue Zhao[1], Yuanjun Xiong[1], Limin Wang[2], Zhirong Wu[1], Xiaoou Tang[1], and Dahua Lin[1]

[1]Department of Information Engineering, The Chinese University of Hong Kong
[2]Computer Vision Laboratory, ETH Zurich, Switzerland

Abstract

Detecting actions in untrimmed videos is an important yet challenging task. In this paper, we present the structured segment network (SSN), a novel framework which models the temporal structure of each action instance via a structured temporal pyramid. On top of the pyramid, we further introduce a decomposed discriminative model comprising two classifiers, respectively for classifying actions and determining completeness. This allows the framework to effectively distinguish positive proposals from background or incomplete ones, thus leading to both accurate recognition and localization. These components are integrated into a unified network that can be efficiently trained in an end-to-end fashion. Additionally, a simple yet effective temporal action proposal scheme, dubbed temporal actionness grouping (TAG) is devised to generate high quality action proposals. On two challenging benchmarks, THUMOS14 and ActivityNet, our method remarkably outperforms previous state-of-the-art methods, demonstrating superior accuracy and strong adaptivity in handling actions with various temporal structures. [1]

1. Introduction

Temporal action detection has drawn increasing attention from the research community, owing to its numerous potential applications in surveillance, video analytics, and other areas [29, 24, 55, 37]. This task is to detect human action instances from untrimmed, and possibly very long videos. Compared to action recognition, it is substantially more challenging, as it is expected to output not only the action category, but also the precise starting and ending time points.

Over the past several years, the advances in convolutional neural networks have led to remarkable progress in video analysis. Notably, the accuracy of action recognition has been significantly improved [39, 44, 9, 49, 51]. Yet, the performances of action detection methods remain unsatisfactory [56, 55, 41]. For existing approaches, one major challenge in precise temporal localization is the large number of incomplete action fragments in the proposed temporal regions. Traditional snippet based classifiers rely on discriminative snippets of actions, which would also exist in these incomplete proposals. This makes them very hard to distinguish from valid detections (see Fig. 1). We argue that tackling this challenge requires the capability of temporal structure analysis, or in other words, the ability to identify different stages *e.g. starting*, *course*, and *ending*, which

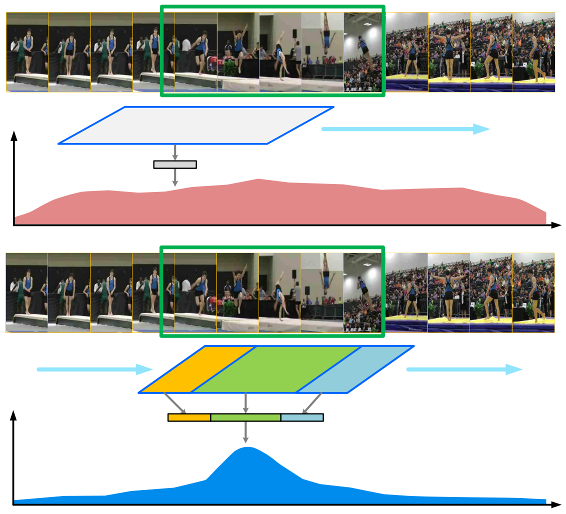

Figure 1. Importance of modeling stage structures in action detection. We slide window detectors through a video clip with an action instance of "Tumbling" (green box). **Top**: The detector builds features without any stage structure of the action, *e.g.* average pooling throughout the window. It produces high responses whenever it sees any discriminative snippet related to tumbling, making it hard to localize the instance. **Bottom**: SSN detector utilizes stage structures (starting, course, and ending) via structured temporal pyramid pooling. Its response is only significant when the window is well aligned.

[1]Code available at http://yjxiong.me/others/ssn

together decide the *completeness* of an actions instance.

Structural analysis is not new in computer vision. It has been well studied in various tasks, *e.g*. image segmentation [20], scene understanding [16], and human pose estimation [1]. Take the most related object detection for example, in deformable part based models (DPM) [8], the modeling of the spatial configurations among parts is crucial. Even with the strong expressive power of convolutional networks [12], explicitly modeling spatial structures, in the form of spatial pyramids [22, 14], remains an effective way to achieve improved performance, as demonstrated in a number of state-of-the-art object detection frameworks, *e.g*. Fast R-CNN [11] and region-based FCN [23].

In the context of video understanding, although temporal structures have played an crucial role in action recognition [28, 48, 32, 52], their modeling in temporal action detection was not as common and successful. Snippet based methods [24, 41] often process individual snippets independently without considering the temporal structures among them. Later works attempt to incorporate temporal structures, but are often limited to analyzing short clips. S-CNN [37] models the temporal structures via the 3D convolution, but its capability is restricted by the underlying architecture [44], which is designed to accommodate only 16 frames. The methods based on recurrent networks [4, 26] rely on dense snippet sampling and thus are confronted with serious computational challenges when modeling long-term structures. Overall, existing works are limited in two key aspects. First, the tremendous amount of visual data in videos restricts their capability of modeling long-term dependencies in an end-to-end manner. Also, they neither provide *explicit* modeling of different stages in an activity (*e.g. starting* and *ending*) nor offer a mechanism to assess the *completeness*, which, as mentioned, is crucial for accurate action detection.

In this work, we aim to move beyond these limitations and develop an effective technique for temporal action detection. Specifically, we adopt the proven paradigm of "proposal+classification", but take a significant step forward by utilizing explicit structural modeling in the temporal dimension. In our model, each complete activity instance is considered as a composition of three major stages, namely *starting*, *course*, and *ending*. We introduce structured temporal pyramid pooling to produce a global representation of the entire proposal. Then we introduce a decomposed discriminative model to jointly classify action categories and determine *completeness* of the proposals, which work collectively to output only complete action instances. These components are integrated into a unified network, called *structured segment network* (SSN). We adopt the sparse snippet sampling strategy [51], which overcomes the computational issue for long-term modeling and enables efficient end-to-end training of SSN. Additionally, we propose to use multi-scale grouping upon the temporal actionness signal to generate action proposals, achieving higher temporal recall with less proposals to further boost the detection performance.

The proposed SSN framework excels in the following aspects: 1) It provides an effective mechanism to model the temporal structures of activities, and thus the capability of discriminating between complete and incomplete proposals. 2) It can be efficiently learned in an end-to-end fashion (5 to 15 hours over a large video dataset, *e.g*. ActivityNet), and once trained, can perform fast inference of temporal structures. 3) The method achieves superior detection performance on standard benchmark datasets, establishing new state-of-the-art for temporal action detection.

2. Related Work

Action Recognition. Action recognition has been extensively studied in the past few years [21, 46, 39, 44, 49, 51, 57]. Earlier methods are mostly based on hand-crafted visual features [21, 46]. In the past several years, the wide adoption of convolutional networks (CNNs) has resulted in remarkable performance gain. CNNs are first introduced to this task in [19]. Later, two-stream architectures [39] and 3D-CNN [44] are proposed to incorporate both appearance and motion features. These methods are primarily frame-based and snippet-based, with simple schemes to aggregate results. There are also efforts that explore long-range temporal structures via temporal pooling or RNNs [49, 27, 4]. However, most methods assume well-trimmed videos, where the action of interest lasts for nearly the entire duration. Hence, they don't need to consider the issue of localizing the action instances.

Object Detection. Our action detection framework is closely related to object detection frameworks [8, 12, 34] in spatial images, where detection is performed by classifying object proposals into foreground classes and a background class. Traditional object proposal methods rely on dense sliding windows [8] and bottom-up methods that exploit low-level boundary cues [45, 58]. Recent proposal methods based on deep neural networks show better average recall while requiring less candidates [34]. Deep models also introduce great modeling capacity for capturing object appearances. With strong visual features, spatial structural modeling [22] remains a key component for detection. In particular, the RoI pooling [11] is introduced to model the spatial configuration of object with minimal extra cost. The idea is further reflected in R-FCN [23] where the spatial configuration is handled with the position sensitive pooling.

Temporal Action Detection. Previous works on activity detection mainly use sliding windows as candidates and focus on designing hand-crafted feature representations for classification [10, 43, 29, 24, 56, 17]. Recent works incorporate deep networks into the detection frameworks and

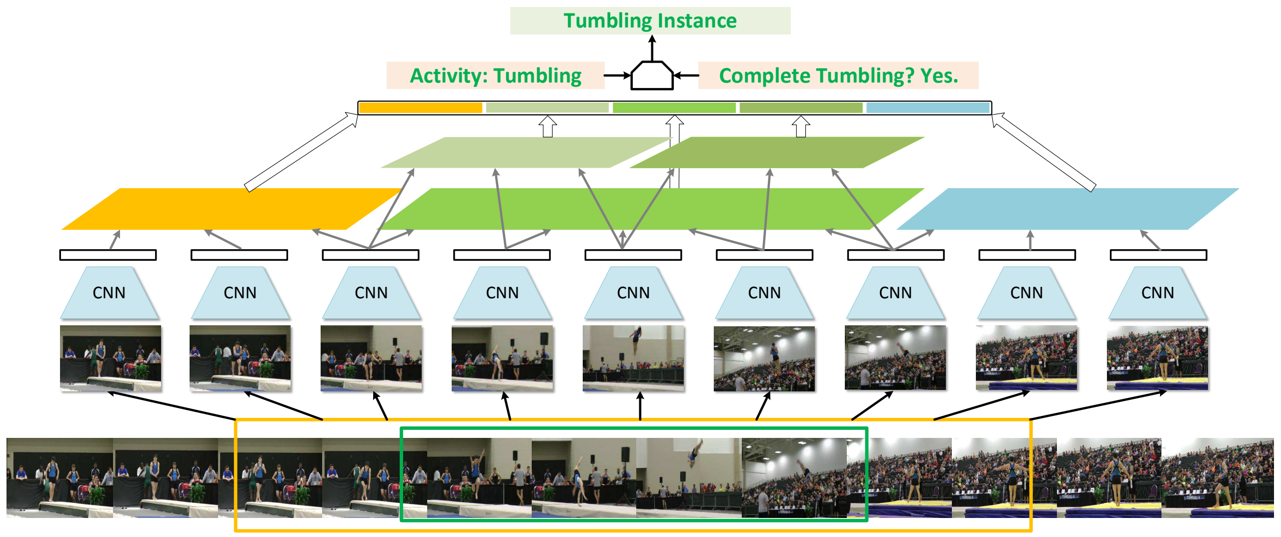

Figure 2. An overview of the structured segment network framework. On a video from ActivityNet [7] there is a candidate region (green box). We first build the augmented proposal (yellow box) by extending it. The augmented proposal is divided into starting (orange), course (green), and ending (blue) stages. An additional level of pyramid with two sub-parts is constructed on the course stage. Features from CNNs are pooled within these five parts and concatenated to form the global region representations. The activity classifier and the completeness classifier operate on the the region representations to produce activity probability and class conditional completeness probability. The final probability of the proposal being positive instance is decided by the joint probability from these two classifiers. During training, we sparsely sample $L = 9$ snippets from evenly divided segments to approximate the dense temporal pyramid pooling.

obtain improved performance [55, 37, 2]. S-CNN [37] proposes a multi-stage CNN which boosts accuracy via a localization network. However, S-CNN relies on C3D [44] as the feature extractor, which is initially designed for snippet-wise action classification. Extending it to detection with possibly long action proposals needs enforcing an undesired large temporal kernel stride. Another work [55] uses Recurrent Neural Network (RNN) to learn a glimpse policy for predicting the starting and ending points of an action. Such sequential prediction is often time-consuming for processing long videos and it does not support joint training of the underlying feature extraction CNN. Our method differs from these approaches in that it explicitly models the action structure via structural temporal pyramid pooling. By using sparse sampling, we further enable efficient end-to-end training. Note there are also works on spatial-temporal detection [13, 54, 25, 50, 31] and temporal video segmentation [15], which are beyond the scope of this paper.

3. Structured Segment Network

The proposed structured segment network framework, as shown in Figure 2, takes as input a video and a set of temporal action proposals. It outputs a set of predicted *activity instances* each associated with a category label and a temporal range (delimited by a starting point and an ending point). From the input to the output, it takes three key steps. First, the framework relies on a proposal method to produce a set of *temporal proposals* of varying durations, where each proposal comes with a starting and an ending time. The proposal methods will be discussed in detail in Section 5. Our framework considers each proposal as a composition of three consecutive stages, *starting*, *course*, and *ending*, which respectively capture how the action starts, proceeds, and ends. Thus upon each proposal, structured temporal pyramid pooling (STPP) are performed by 1) splitting the proposal into the three stages; 2) building temporal pyramidal representation for each stage; 3) building global representation for the whole proposal by concatenating stage-level representations. Finally, two classifiers respectively for recognizing the activity category and assessing the completeness will be applied on the representation obtained by STPP and their predictions will be combined, resulting in a subset of *complete* instances tagged with category labels. Other proposals, which are considered as either *belonging to background* or *incomplete*, will be filtered out. All the components outlined above are integrated into a unified network, which will be trained in an end-to-end way. For training, we adopt the sparse snippet sampling strategy [51] to approximate the temporal pyramid on dense samples. By exploiting the redundancy among video snippets, this strategy can substantially reduce the computational cost, thus allowing the crucial modeling of long-term temporal structures.

3.1. Three-Stage Structures

At the input level, a video can be represented as a sequence of T *snippets*, denoted as $(S_t)_{t=1}^{T}$. Here, one snippet contains several consecutive frames, which, as a whole, is characterized by a combination of RGB images and an optical flow stack [39]. Consider a given set of N proposals $P = \{p_i = [s_i, e_i]\}_{i=1}^N$. Each proposal p_i is composed of a starting time s_i and an ending time e_i. The duration of p_i is thus $d_i = e_i - s_i$. To allow structural analysis and particularly to determine whether a proposal captures a *complete* instance, we need to put it in a context. Hence, we augment each proposal p_i into $p'_i = [s'_i, e'_i]$ with where $s'_i = s_i - d_i/2$ and $e'_i = e_i + d_i/2$. In other words, the augmented proposal p'_i doubles the span of p_i by extending beyond the starting and ending points, respectively by $d_i/2$. If a proposal accurately aligns well with a groundtruth instance, the augmented proposal will capture not only the inherent process of the activity, but also how it starts and ends. Following the three-stage notion, we divide the augmented proposal p'_i into three consecutive intervals: $p_i^s = [s'_i, s_i]$, $p_i^c = [s_i, e_i]$, and $p_i^e = [e_i, e'_i]$, which are respectively corresponding to the *starting*, *course*, and *ending* stages.

3.2. Structured Temporal Pyramid Pooling

As mentioned, the structured segment network framework derives a global representation for each proposal via temporal pyramid pooling. This design is inspired by the success of spatial pyramid pooling [22, 14] in object recognition and scene classification. Specifically, given an augmented proposal p'_i divided into three stages p_i^s, p_i^c, and p_i^e, we first compute the stage-wise feature vectors $\mathbf{f}_i^s$, $\mathbf{f}_i^c$, and $\mathbf{f}_i^e$ respectively via temporal pyramid pooling, and then concatenate them into a global representation.

Specifically, a stage with interval $[s, e]$ would cover a series of snippets, denoted as $\{S_t | s \leq t \leq e\}$. For each snippet, we can obtain a feature vector $\mathbf{v}_t$. Note that we can use any feature extractor here. In this work, we adopt the effective two-stream feature representation first proposed in [39]. Based on these features, we construct a L-level temporal pyramid where each level evenly divides the interval into B_l parts. For the i-th part of the l-th level, whose interval is $[s_{li}, e_{li}]$, we can derive a pooled feature as

$$\mathbf{u}_i^{(l)} = \frac{1}{|e_{li} - s_{li} + 1|} \sum_{t=s_{li}}^{e_{li}} \mathbf{v}_t. \tag{1}$$

Then the overall representation of this stage can be obtained by concatenating the pooled features across all parts at all levels as $\mathbf{f}_i^c = (\mathbf{u}_i^{(l)} | l = 1, \ldots, L, \ i = 1, \ \ldots, B_l)$.

We treat the three stages differently. Generally, we observed that the *course* stage, which reflects the activity process itself, usually contains richer structure *e.g.* this process itself may contain sub-stages. Hence, we use a two-level pyramid, *i.e.* $L = 2, B_1 = 1$, and $B_2 = 2$, for the *course* stage, while using simpler one-level pyramids (which essentially reduce to standard average pooling) for *starting* and *ending* pyramids. We found empirically that this setting strikes a good balance between expressive power and complexity. Finally, the stage-wise features are combined via concatenation. Overall, this construction explicitly leverages the structure of an activity instance and its surrounding context, and thus we call it *structured temporal pyramid pooling* (STPP).

3.3. Activity and Completeness Classifiers

On top of the structured features described above, we introduce two types of classifiers, an *activity classifier* and a set of *completeness classifiers*. Specifically, the *activity classifier* A classifies input proposals into $K + 1$ classes, *i.e.* K activity classes (with labels $1, \ldots, K$) and an additional *"background"* class (with label 0). This classifier restricts its scope to the *course* stage, making predictions based on the corresponding feature $\mathbf{f}_i^c$. The *completeness classifiers* $\{C_k\}_{k=1}^K$ are a set of binary classifiers, each for one activity class. Particularly, C_k predicts whether a proposal captures a *complete* activity instance of class k, based on the global representation $\{\mathbf{f}_i^s, \mathbf{f}_i^c, \mathbf{f}_i^e\}$ induced by STPP. In this way, the *completeness* is determined not only on the proposal itself but also on its surrounding context.

Both types of classifiers are implemented as linear classifiers on top of high-level features. Given a proposal p_i, the activity classifier will produce a vector of normalized responses via a softmax layer. From a probabilistic view, it can be considered as a conditional distribution $P(c_i | p_i)$, where c_i is the class label. For each activity class k, the corresponding completeness classifier C_k will yield a probability value, which can be understood as the conditional probability $P(b_i | c_i, p_i)$, where b_i indicates whether p_i is *complete*. Both outputs together form a joint distribution. When $c_i \geq 1$, $P(c_i, b_i | p_i) = P(c_i | p_i) \cdot P(b_i | c_i, p_i)$. Hence, we can define a *unified classification loss* jointly on both types of classifiers. With a proposal p_i and its label c_i:

$$\mathcal{L}_{cls}(c_i, b_i; p_i) = -\log P(c_i | p_i) - 1_{(c_i \geq 1)} \log P(b_i | c_i, p_i). \tag{2}$$

Here, the *completeness* term $P(b_i | c_i, p_i)$ is only used when $c_i \geq 1$, *i.e.* the proposal p_i is not considered as part of the background. Note that these classifiers together with STPP are integrated into a single network that is trained in an *end-to-end* way.

During training, we collect three types of proposal samples: (1) *positive proposals*, *i.e.* those overlap with the closest groundtruth instances with at least 0.7 IoU; (2) *background proposals*, *i.e.* those that do not overlap with any groundtruth instances; and (3) *incomplete proposals*, *i.e.* those that satisfy the following criteria: 80% of its own span is contained in a groundtruth instance, while its IoU with that instance is below 0.3 (in other words, it just covers a small part of the instance). For these proposal types, we respectively have $(c_i > 0, b_i = 1)$, $c_i = 0$, and

$(c_i > 0, b_i = 0)$. Each mini-batch is ensured to contain all three types of proposals.

3.4. Location Regression and Multi-Task Loss

With the structured information encoded in the global features, we can not only make categorical predictions, but also refine the proposal's temporal interval itself by *location regression*. We devise a set of location regressors $\{R_k\}_{k=1}^K$, each for an activity class. We follow the design in RCN-N [12], but adapting it for 1D temporal regions. Particularly, for a *positive proposal* p_i, we regress the relative changes of both the interval center μ_i and the span ϕ_i (in log-scale), using the closest groundtruth instance as the target. With both the classifiers and location regressors, we define a multi-task loss over an training sample p_i, as:

$$\mathcal{L}_{cls}(c_i, b_i; p_i) + \lambda \cdot 1_{(c_i \geq 1 \ \& \ b_i = 1)} \mathcal{L}_{reg}(\mu_i, \phi_i; p_i). \quad (3)$$

Here, $\mathcal{L}_{reg}$ uses the smooth L_1 loss function [11].

4. Efficient Training and Inference with SSN

The huge amount of frames poses a serious challenge in computational cost to video analysis. Our structured segment network also faces this challenge. This section presents two techniques which we use to reduce the cost and enable end-to-end training.

Training with sparse sampling. The structured temporal pyramid, in its original form, rely on densely sampled snippets. This would lead to excessive computational cost and memory demand in end-to-end training over long proposals – in practice, proposals that span over hundreds of frames are not uncommon. However, dense sampling is generally unnecessary in our framework. Particularly, the *pooling* operation is essentially to collect feature statistics over a certain region. Such statistics can be well approximated via a subset of snippets, due to the high redundancy among them.

Motivated by this, we devise a *sparse snippet sampling scheme*. Specifically, given a augmented proposal p_i', we evenly divide it into $L = 9$ segments, randomly sampling only one snippet from each segment. Structured temporal pyramid pooling is performed for each pooling region on its corresponding segments. This scheme is inspired by the segmental architecture in [51], but differs in that it operates within STPP instead of a global average pooling. In this way, we fix the number of features needed to be computed regardless of how long the proposal is, thus effectively reducing the computational cost, especially for modeling long-term structures. More importantly, this enables end-to-end training of the entire framework over a large number of long proposals.

Inference with reordered computation. In testing, we sample video snippets with a fixed interval of 6 frames, and construct the temporal pyramid thereon. The original formulation of temporal pyramid first computes pooled features and then applies the classifiers and regressors on top which is not efficient. Actually, for each video, hundreds of proposals will be generated, and these proposals can significantly overlap with each other – therefore, a considerable portion of the snippets and the features derived thereon are shared among proposals.

To exploit this redundancy in the computation, we adopt the idea introduced in position sensitive pooling [23] to improve testing efficiency. Note that our classifiers and regressors are both linear. So the key step in classification or regression is to multiply a weight matrix $\mathbf{W}$ with the global feature vector $\mathbf{f}$. Recall that $\mathbf{f}$ itself is a concatenation of multiple features, each pooled over a certain interval. Hence the computation can be written as $\mathbf{Wf} = \sum_j \mathbf{W}_j \mathbf{f}_j$, where j indexes different regions along the pyramid. Here, $\mathbf{f}_j$ is obtained by *average pooling* over all snippet-wise features within the region r_j. Thus, we have

$$\mathbf{W}_j \mathbf{f}_j = \mathbf{W}_j \cdot \mathbb{E}_{t \sim r_j}[\mathbf{v}_t] = \mathbb{E}_{t \sim r_j}[\mathbf{W}_j \mathbf{v}_t]. \quad (4)$$

$\mathbb{E}_{t \sim r_j}$ denotes the average pooling over r_j, which is a linear operation and therefore can be exchanged with the matrix multiplication. Eq (4) suggests that the linear responses w.r.t. the classifiers/regressors can be computed *before* pooling. In this way, the heavy matrix multiplication can be done in the CNN for each video over all snippets, and for each proposal, we only have to pool over the network outputs. This technique can reduce the processing time after extracting network outputs from around 10 seconds to less than 0.5 second per video on average.

5. Temporal Region Proposals

In general, SSN accepts arbitrary proposals, *e.g.* sliding windows [37, 56]. Yet, an effective proposal method can produce more accurate proposals, and thus allowing a small number of proposals to reach a certain level of performance. In this work, we devise an effective proposal method called *temporal actionness grouping (TAG)*.

This method uses an actionness classifier to evaluate the binary *actionness probabilities* for individual snippets. The use of binary actionness for proposals is first introduced in spatial action detection by [50]. Here we utilize it for temporal action detection.

Our basic idea is to find those continuous temporal regions with mostly high actionness snippets to serve as proposals. To this end, we repurpose a classic watershed algorithm [36], applying it to the 1D signal formed by a sequence of *complemented* actionness values, as shown in Figure 3. Imagine the signal as 1D terrain with heights and basins. This algorithm floods water on this terrain with different *"water level"* (γ), resulting in a set of *"basins"* covered by water, denoted by $G(\gamma)$. Intuitively, each "basin"

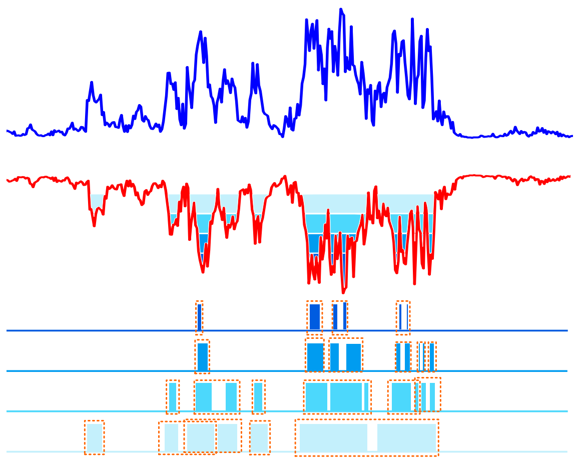

Figure 3. Visualization of the temporal actionness grouping process for proposal generation. **Top**: Actionness probabilities as a 1D signal sequence. **Middle**: The complement signal. We flood it with different levels γ. **Bottom**: Regions obtained by different flooding levels. By merging the regions according to the grouping criterion, we get the final set of proposals (in orange color).

corresponds to a temporal region with high actionness. The ridges above water then form the blank areas between basins, as illustrated in Fig. 3.

Given a set of basins $G(\gamma)$, we devise a grouping scheme similar to [33], which tries to connect small basins into proposal regions. The scheme works as follows: it begins with a seed basin, and consecutively absorbs the basins that follow, until the fraction of the basin durations over the total duration (*i.e.* from the beginning of the first basin to the ending of the last) drops below a certain threshold τ. The absorbed basins and the blank spaces between them are then grouped to form a single proposal. We treat each basin as seed and perform the grouping procedure to obtain a set of proposals denoted by $G'(\tau, \gamma)$. Note that we do not choose a specific combination of τ and γ. Instead we uniformly sample τ and γ from $\in (0, 1)$ with an even step of 0.05. The combination of these two thresholds leads to multiple sets of regions. We then take the *union* of them. Finally, we apply non-maximal suppression to the union with IoU threshold 0.95, to filter out highly overlapped proposals. The retained proposals will be fed to the SSN framework.

6. Experimental Results

We conducted experiments to test the proposed framework on two large-scale action detection benchmark datasets: *ActivityNet* [7] and *THUMOS14* [18]. In this section we first introduce these datasets and other experimental settings and then investigate the impact of different components via a set of ablation studies. Finally we compare the performance of SSN with other state-of-the-art approaches.

6.1. Experimental Settings

Datasets. ActivityNet [7] has two versions, *v1.2* and *v1.3*. The former contains 9682 videos in 100 classes, while the latter, which is a superset of v1.2 and was used in the ActivityNet Challenge 2016, contains 19994 videos in 200 classes. In each version, the dataset is divided into three disjoint subsets, training, validation, and testing, by 2:1:1. **THUMOS14** [18] has 1010 videos for validation and 1574 videos for testing. This dataset does not provide the training set by itself. Instead, the UCF101 [42], a trimmed video dataset is appointed as the official training set. Following the standard practice, we train out models on the validation set and evaluate them on the testing set. On these two sets, 220 and 212 videos have temporal annotations in 20 classes, respectively. 2 falsely annotated videos ("270","1496") in the test set are excluded in evaluation. In our experiments, we compare with our method with the states of the art on both *THUMOS14* and *ActivityNet v1.3*, and perform ablation studies on *ActivityNet v1.2*.

Implementation Details. We train the structured segment network in an end-to-end manner, with raw video frames and action proposals as the input. Two-stream CNNs [39] are used for feature extraction. We also use the spatial and temporal streams to harness both the appearance and motion features. The binary actionness classifiers underlying the TAG proposals are trained with [51] on the training subset of each dataset. We use SGD to learn CNN parameters in our framework, with batch size 128 and momentum 0.9. We initialize the CNNs with pre-trained models from ImageNet [3]. The initial learning rates are set to 0.001 for RGB networks and 0.005 for optical flow networks. In each minibatch, we keep the ratio of three types of proposals, namely *positive*, *background*, and *incomplete*, to be 1:1:6. For the completeness classifiers, only the samples with loss values ranked in the first $1/6$ of a minibatch are used for calculating gradients, which resembles online hard negative mining [38]. On both versions of ActivityNet, the RGB and optical flow branches of the two-stream CNN are respectively trained for $9.5K$ and $20K$ iterations, with learning rates scaled down by 0.1 after every $4K$ and $8K$ iterations, respectively. On THUMOS14, these two branches are respectively trained for $1K$ and $6K$ iterations, with learning rates scaled down by 0.1 per 400 and 2500 iterations.

Evaluation Metrics. As both datasets originate from contests, each dataset has its own convention of reporting performance metrics. We follow their conventions, reporting mean average precision (mAP) at different IoU thresholds. On both versions of ActivityNet, the IoU thresholds are $\{0.5, 0.75, 0.95\}$. The average of mAP values with IoU thresholds $[0.5{:}0.05{:}0.95]$ is used to compare the performance between different methods. On THUMOS14, the IoU thresholds are $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. The mAP at 0.5

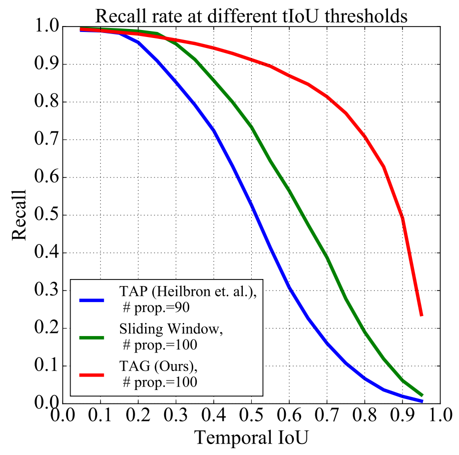

Figure 4. Recall rate at different tIoU thresholds on ActivityNet v1.2. High recall rates at high IoU thresholds (> 0.7) indicate better proposal quality.

Proposal Method	THUMOS14		ActivityNet v1.2	
	# Prop.	AR	# Prop.	AR
Sliding Windows	204	21.2	100	34.8
SCNN-prop [37]	200	20.0	-	-
TAP [6]	200	23.0	90	14.9
DAP [5]	200	37.0	100	12.1
TAG	200	**48.9**	100	**71.7**

Table 1. Comparison between different temporal action proposal methods with same number of proposals. "AR" refers to the average recall rates. "-" indicates the result is not available.

IoU is used for comparing results from different methods.

6.2. Ablation Studies

Temporal Action Proposal. We compare the performance of different action proposal schemes in three aspects, *i.e.* recall, quality, and detection performance. Particularly, we compare our TAG scheme with common sliding windows as well as other state-of-the-art proposal methods, including SCNN-prop, a proposal networks presented in [37], TAP [6], DAP [5]. For the sliding window scheme, we use 20 exponential scales starting from 0.3 second long and step sizes of 0.4 times of window lengths.

We first evaluate the average recall rates, which are summarized in Table 1. We can see that TAG proposal have higher recall rates with the same number of proposals. Then we investigate the quality of its proposals. We plot the recall rates from different proposal methods at different IoU thresholds in Fig. 4. We can see TAG retains relatively high recall at high IoU thresholds, demonstrating that the proposals from TAG are generally more accurate. In experiments we also tried applying the actionness classifier trained on ActivityNet v1.2 directly on THUMOS14. We can still achieve a reasonable average recall of 39.6%, while the one trained on THUMOS14 achieves 48.9% in Table 1. Finally, we evaluate the proposal methods in the context of action detection. The detection mAP values using sliding window proposals and TAG proposals are shown in Table 3. The results confirm that, in most cases, the improved proposals can result in improved detection performance.

Average mAP (%)	(1)-0	(1,2)-0	(1)-1	(1,2)-1
Max Pool	13.1	13.5	18.3	18.4
Average Pool	4.48	4.34	24.3	24.6

Table 2. Comparison between different temporal pooling settings. The setting (1,2)-1 is used in the SSN framework. Please refer to Sec. 6.2 for the definition of these settings.

Structured Temporal Pyramid Pooling. Here we study the influence of different pooling strategies in STPP. We denote one pooling configuration as $(B_1, \dots, B_K) - A$, where K refers to the number of pyramid levels for the course stage and $B_1, \dots, B_K$ the number of regions in each level. $A = 1$ indicates we use augmented proposal and model the starting and ending stage, while $A = 0$ indicates we only use the original proposal (without augmentation). Additionally we compare two within-region pooling methods: average and max pooling. The results are summarized in Table 2. Note that these configurations are evaluated in the stage-wise training scenario. We observe that cases where $A = 0$ have inferior performance, showing that the introduction of the stage structure is very important for accurate detection. Also, increasing the depth of the pyramids for the course stage can give slight performance gain. Based on these results, we fix the configuration to $(1, 2) - 1$ in later experiments.

Classifier Design. In this work, we introduced the activity and completeness classifiers which work together to classify the proposal. We verify the importance of this decomposed design by studying another design that replaces it with a single set of classifiers, for which both *background* and *incomplete* samples are uniformly treated as negative. We perform similar negative sample mining for this setting. The results are summarized in Table 3. We observe that using only one classifier to distinguish *positive* samples from both *background* and *incomplete* would lead to worse result even with negative mining, where mAP decreased from 23.7% to 17.9%. We attribute this performance gain to the different natures of the two negative proposal types, which require different classifiers to handle.

Location Regression & Multi-Task Learning. Because of the contextual information contained in the starting and ending stages of the global region features, we are able to perform location regression. We measure the contribution of this step to the detection performance in Table 3. From the results we can see that the location regression and multi-task learning, where we train the classifiers and the regres-

	Stage-Wise				End-to-End	
STPP		✓	✓	✓	✓	✓
Act. + Comp.			✓	✓	✓	✓
Loc. Reg.				✓		✓
SW	0.56	5.99	16.4	18.1	-	-
TAG	4.82	17.9	24.6	24.9	24.8	25.9

Table 3. Ablation study on ActivityNet [7] v1.2. Overall, end-to-end training is compared against stage wise training. We evaluate the performance using both sliding window proposals ("SW") and TAG proposals ("TAG"), measured by mean average precision (mAP). Here, "STPP" refers to structure temporal pyramid pooling. "Act. + Comp." refers to the use of two classifiers design. "Loc. Reg" denotes the use the location regression.

sors together in an end-to-end manner, always improve the detection accuracy.

Training: Stage-wise v.s. End-to-end. While the structured segment network is designed for end-to-end training, it is also possible to first densely extract features and train the classifiers and regressors with SVM and ridge regression, respectively. We refer to this training scheme as stage-wise training. We compare the performance of end-to-end training and stage-wise training in Table 3. We observe that models from end-to-end training can slightly outperform those learned with stage-wise training under the same settings. This is remarkable as we are only sparsely sampling snippets in end-to-end training, which also demonstrates the importance of jointly optimizing the classifiers and feature extractors and justifies our framework design. Besides, end-to-end training has another major advantage that it does not need to store the extracted features for the training set, which could become quite storage intensive as training data grows.

6.3. Comparison with the State of the Art

Finally, we compare our method with other state-of-the-art temporal action detection methods on THUMOS14 [18] and ActivityNet v1.3 [7], and report the performances using the metrics described above. Note that the average action duration in THUMOS14 and ActivityNet are 4 and 50 seconds. And the average video duration are 233 and 114 seconds, respectively. This reflects the distinct natures of these datasets in terms of the granularities and temporal structures of the action instances. Hence, strong adaptivity is required to perform consistently well on both datasets.

THUMOS14. On THUMOS 14, We compare with the contest results [47, 30, 35] and those from recent works, including the methods that use segment-based 3D CNN [37], score pyramids [56], and recurrent reinforcement learning [55]. The results are shown in Table 4. In most cases, the proposed method outperforms previous state-of-the-art methods by over 10% in absolute mAP values.

THUMOS14, mAP@α					
Method	0.1	0.2	0.3	0.4	0.5
Wang *et. al.* [47]	18.2	17.0	14.0	11.7	8.3
Oneata *et. al.* [30]	36.6	33.6	27.0	20.8	14.4
Richard *et. al.* [35]	39.7	35.7	30.0	23.2	15.2
S-CNN [37]	47.7	43.5	36.3	28.7	19.0
Yeung *et. al.* [55]	48.9	44.0	36.0	26.4	17.1
Yuan *et. al.* [56]	51.4	42.6	33.6	26.1	18.8
SSN	**60.3**	**56.2**	**50.6**	**40.8**	**29.1**
SSN*	**66.0**	**59.4**	**51.9**	**41.0**	**29.8**

Table 4. Action detection results on THUMOS14, measured by mAP at different IoU thresholds α. The upper half of the table shows challenge results back in 2014. "SSN*" indicates metrics calculated in the PASCAL-VOC style used by ActivityNet [7].

ActivityNet v1.3 (testing), **mAP@α**				
Method	0.5	0.75	0.95	Average
Wang *et. al.* [53]	42.48	2.88	0.06	14.62
Singh *et. al.* [40]	28.67	17.78	2.88	17.68
Singh *et. al.* [41]	36.40	11.05	0.14	17.83
SSN	**43.26**	**28.70**	**5.63**	**28.28**

Table 5. Action detection results on ActivityNet v1.3, measured by mean average precision (mAP) for different IoU thresholds α and the average mAP of IoU thresholds from 0.5 to 0.95.

ActivityNet. The results on the testing set of ActivityNet v1.3 are shown in Table 5. For references, we list the performances of highest ranked entries in the ActivityNet 2016 challenge. We submit our results to the test server of ActivityNet v1.3 and report the detection performance on the testing set. The proposed framework, using a single model instead of an ensemble, is able to achieve an average mAP of 28.28 and perform well at high IOU thresholds, *i.e.*, 0.75 and 0.95. This clearly demonstrates the superiority of our method.

7. Conclusion

In this paper, we presented a generic framework for temporal action detection, which combines a structured temporal pyramid with two types of classifiers, respectively for predicting activity class and completeness. With this framework, we achieved significant performance gain over state-of-the-art methods on both ActivityNet and THUMOS14. Moreover, we demonstrated that our method is both accurate and generic, being able to localize temporal boundaries precisely and working well for activity classes with very different temporal structures.

Acknowledgment. This work is partially supported by the Big Data Collaboration Research grant from SenseTime Group (CUHK Agreement No. TS1610626), the General Research Fund (GRF) of Hong Kong (No. 14236516) and the Early Career Scheme (ECS) of Hong Kong (No. 24204215).

References

[1] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *CVPR*, pages 1014–1021. IEEE, 2009. 2

[2] R. De Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, and T. Tuytelaars. Online action detection. In *ECCV*, pages 269–284. Springer, 2016. 3

[3] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. ImageNet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 6

[4] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, pages 2625–2634, 2015. 2

[5] V. Escorcia, F. Caba Heilbron, J. C. Niebles, and B. Ghanem. Daps: Deep action proposals for action understanding. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *ECCV*, pages 768–784, 2016. 7

[6] B. G. Fabian Caba Heilbron, Juan Carlos Niebles. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *CVPR*, pages 1914–1923, 2016. 7

[7] B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, pages 961–970, 2015. 3, 6, 8

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE TPAMI*, 32(9):1627–1645, 2010. 2

[9] B. Fernando, E. Gavves, J. O. M., A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *CVPR*, pages 5378–5387, 2015. 1

[10] A. Gaidon, Z. Harchaoui, and C. Schmid. Temporal localization of actions with actoms. *IEEE TPAMI*, 35(11):2782–2795, 2013. 2

[11] R. Girshick. Fast r-cnn. In *ICCV*, pages 1440–1448, 2015. 2, 5

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014. 2, 5

[13] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, pages 759–768, June 2015. 3

[14] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, pages 346–361. Springer, 2014. 2, 4

[15] M. Hoai, Z.-Z. Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *CVPR*, pages 3265–3272. IEEE, 2011. 3

[16] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *IJCV*, 80(1):3–15, 2008. 2

[17] M. Jain, J. C. van Gemert, H. Jégou, P. Bouthemy, and C. G. M. Snoek. Action localization by tubelets from motion. In *CVPR*, June 2014. 2

[18] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/THUMOS14/`, 2014. 6, 8

[19] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, pages 1725–1732, 2014. 2

[20] J. Lafferty, A. McCallum, F. Pereira, et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, volume 1, pages 282–289, 2001. 2

[21] I. Laptev. On space-time interest points. *IJCV*, 64(2-3), 2005. 2

[22] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, pages 2169–2178. IEEE, 2006. 2, 4

[23] Y. Li, K. He, J. Sun, et al. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, pages 379–387, 2016. 2, 5

[24] P. Mettes, J. C. van Gemert, S. Cappallo, T. Mensink, and C. G. Snoek. Bag-of-fragments: Selecting and encoding video fragments for event detection and recounting. In *ICMR*, pages 427–434, 2015. 1, 2

[25] P. Mettes, J. C. van Gemert, and C. G. Snoek. Spot on: Action localization from pointly-supervised proposals. In *ECCV*, pages 437–453. Springer, 2016. 3

[26] A. Montes, A. Salvador, S. Pascual, and X. Giro-i Nieto. Temporal activity detection in untrimmed videos with recurrent neural networks. In *NIPS Workshop*, 2016. 2

[27] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, pages 4694–4702, 2015. 2

[28] J. C. Niebles, C.-W. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, pages 392–405. Springer, 2010. 2

[29] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, pages 1817–1824, 2013. 1, 2

[30] D. Oneata, J. Verbeek, and C. Schmid. The lear submission at thumos 2014. In *THUMOS Action Recognition Challenge*, 2014. 8

[31] X. Peng and C. Schmid. Multi-region two-stream r-cnn for action detection. In *ECCV*. Springer, 2016. 3

[32] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *CVPR*, pages 612–619, 2014. 2

[33] J. Pont-Tuset, P. Arbeláez, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. In *arXiv:1503.00848*, March 2015. 6

[34] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 2

[35] A. Richard and J. Gall. Temporal action detection using a statistical language model. In *CVPR*, pages 3131–3140, 2016. 8

[36] J. B. Roerdink and A. Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta informaticae*, 41(1, 2):187–228, 2000. 5

[37] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage CNNs. In *CVPR*, pages 1049–1058, 2016. 1, 2, 3, 5, 7, 8

[38] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, pages 761–769, 2016. 6

[39] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, pages 568–576, 2014. 1, 2, 4, 6

[40] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *CVPR*, pages 1961–1970, 2016. 8

[41] G. Singh and F. Cuzzolin. Untrimmed video classification for activity detection: submission to activitynet challenge. *CoRR*, abs/1607.01979, 2016. 1, 2, 8

[42] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1212.0402*, 2012. 6

[43] K. Tang, B. Yao, L. Fei-Fei, and D. Koller. Combining the right features for complex event recognition. In *CVPR*, pages 2696–2703, 2013. 2

[44] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, pages 4489–4497, 2015. 1, 2, 3

[45] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders. Segmentation as selective search for object recognition. In *ICCV*, pages 1879–1886, 2011. 2

[46] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, pages 3551–3558, 2013. 2

[47] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. In *THUMOS Action Recognition Challenge*, 2014. 8

[48] L. Wang, Y. Qiao, and X. Tang. Latent hierarchical model of temporal structure for complex activity classification. *IEEE TIP*, 23(2):810–822, 2014. 2

[49] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, pages 4305–4314, 2015. 1, 2

[50] L. Wang, Y. Qiao, X. Tang, and L. Van Gool. Actionness estimation using hybrid fully convolutional networks. In *CVPR*, pages 2708–2717, 2016. 3, 5

[51] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, pages 20–36, 2016. 1, 2, 3, 5, 6

[52] P. Wang, Y. Cao, C. Shen, L. Liu, and H. T. Shen. Temporal pyramid pooling based convolutional neural network for action recognition. *IEEE TCSVT*, 2016. 2

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

[54] P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. In *ICCV*, pages 3164–3172, 2015. 3

[55] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, pages 2678–2687, 2016. 1, 3, 8

[56] J. Yuan, B. Ni, X. Yang, and A. A. Kassim. Temporal action localization with pyramid of score distribution features. In *CVPR*, pages 3093–3102, 2016. 1, 2, 5, 8

[57] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-time action recognition with enhanced motion vector CNNs. In *CVPR*, pages 2718–2726, 2016. 2

[58] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, pages 391–405, 2014. 2