Action Temporal Localization in Untrimmed Videos via Multi-stage CNNs

Zheng Shou, Dongang Wang, and Shih-Fu Chang
Columbia University
New York, NY, USA
{zs2262,dw2648,sc250}@columbia.edu

Abstract

We address action temporal localization in untrimmed long videos. This is important because videos in real applications are usually unconstrained and contain multiple action instances plus video content of background scenes or other activities. To address this challenging issue, we exploit the effectiveness of deep networks in action temporal localization via multi-stage segment-based 3D ConvNets: (1) a proposal stage identifies candidate segments in a long video that may contain actions; (2) a classification stage learns one-vs-all action classification model to serve as initialization for the localization stage; and (3) a localization stage fine-tunes on the model learnt in the classification stage to localize each action instance. We propose a novel loss function for the localization stage to explicitly consider temporal overlap and therefore achieve high temporal localization accuracy. On two large-scale benchmarks, our approach achieves significantly superior performances compared with other state-of-the-art systems: mAP increases from 1.7% to 7.4% on MEXaction2 and increased from 15.0% to 19.0% on THUMOS 2014, when the overlap threshold for evaluation is set to 0.5.

1. Introduction

Impressive progress has been reported in recent literature for action recognition [42, 27, 2, 3, 39, 40, 23, 17, 30, 44, 12, 36]. Besides detecting action in manually trimmed short video, researchers start to develop techniques for detecting actions in untrimmed long videos in the wild. This trend motivates another challenging topic - action temporal localization: given a long untrimmed video, “when does a specific action start and end?” This problem is important because real applications usually involve long untrimmed videos, which can be highly unconstrained in space and time, and one video can contain multiple action instances plus background scenes or other activities. Localizing actions in long videos, such as those in surveillance, can save tremendous time and computational costs.

Most state-of-the-art methods rely on manually selected features, and their performances still require much improvement. For example, top performing approaches in THUMOS Challenge 2014 [26, 41, 16, 14] and 2015 [46, 9] both used improved Dense Trajectory (iDT) with Fisher Vector (FV) [40, 24]. There have been some recent attempts at incorporating iDT features with appearance features automatically extracted by frame-level deep networks [26, 41, 16]. Nevertheless, such 2D ConvNets does not work well because it does not capture motion information, which is important for modeling actions and determining their temporal boundaries.

As an analogy in still images, object detection recently achieved large improvements by using deep networks. Inspired by Region-based Convolutional Neural Networks (R-CNN) [7] and its upgraded versions [6, 29, 20], we develop Segment-CNNs, which is an effective deep network framework for action temporal localization as outlined in Figure 1. Segment-CNNs adopts 3D ConvNets [12, 36], which recently has been shown to be promising for capturing motion characteristics in videos, and add a new multi-stage framework. First, multi-scale segments are generated as candidates for the deep networks of three stages. The proposal stage classifies each segment as either action or background in order to eliminate background segment estimated to be unlikely to contain actions of interest. The classification stage trains typical one-vs-all classification model for all action categories as well as the background.

However, the classification model aims at finding key evidences to distinguish different categories, rather than localizing precise action occurrences in time. Sometimes, the scores from the classification model can be high even when the segment has only a very small overlap with the ground truth instance. This can be detrimental because subsequent post-processing steps, such as non-maximum suppression, might remove segment of small score but large overlap with ground truth. To explicitly take temporal overlap into consideration, we introduce the localization stage based on the same architecture, but this stage uses a novel loss function, which rewards segments with higher temporal overlap with
the ground truths, and thus can generate confidence scores more suitable for post-processing. Note that the classification stage cannot be replaced by localization. We will show later that using the model learnt in the classification stage (without considering temporal overlap) to initialize the model for the localization stage (take into account temporal overlap) is important, and achieves better temporal localization accuracies.

To summarize, our main contributions are three-fold:

1. To the best of our knowledge, our work is the first to exploit 3D ConvNets with multi-stage processes for action temporal localization in untrimmed long videos in the wild.

2. We introduce an effective multi-stage Segment-CNNs framework, to propose candidate segments, recognize actions, and localize temporal boundaries. The first proposal stage improves the efficiency by eliminating unlikely candidate segments, and the last stage on localization is key to temporal localization accuracy boosting.

3. The proposed techniques significantly outperform the state-of-the-art systems over two large-scale benchmarks suitable for action temporal localization. When the overlap threshold used in evaluation is set to 0.5, our approach significantly improves mAP on MEXAction2 from 1.7% to 7.4% and mAP on THUMOS 2014 from 15.0% to 19.0%. We did not evaluate on THUMOS Challenge 2015 [9] because the ground truth is still withheld by organizers for future evaluation. More detailed evaluation results are available in Section 4.

2. Related work

Action temporal localization. This topic has been studied in two directions. When training data only have video-level category labels but no temporal annotations, researchers formulated this as weakly supervised problems or multiple instance learning problems to learn the key evidences in untrimmed videos and temporally localize actions by selecting key instances [21, 22]. Sun et al. [35] transferred knowledge from web images to address temporal localization in untrimmed web videos.

Another line of work focuses on learning from data when the temporal boundaries have been annotated in untrimmed videos, such as THUMOS. Most of these works pose this as a classification problem and adopt a temporal sliding window approach, where each window is considered as an action candidate subject to classification [24]. Surveys about action classification methods can be found in [42, 27, 2, 3], and recently, two directions lead the state-of-the-art: (1) Wang et al. [39] proposed extracting HOG, HOF, MBH features along dense trajectories, and later on they took camera motion into consideration [40]. Further improvement can be achieved by stacking features with multiple time skips [23]. (2) Enlightened by the suc-
cess of CNNs in recent works [19, 31], Karpathy et al. [17] evaluated frame-level CNNs on large-scale video classification tasks. Simonyan and Zisserman [30] designed two-stream CNNs to learn from still image and motion flow respectively. In [44], a latent concept descriptor of convolutional feature map was proposed, and great results were achieved on event detection with VLAD encoding. To learn spatio-temporal features together, the architecture of 3D ConvNets was explored in [12, 36], achieving competitive results. Oneata et al. [25] proposed approximately normalized Fisher Vectors to reduce the high dimensionality of FV. Stoian et al. [34] introduced a two-level cascade to allow fast search for action instances. Instead of precision, these methods focus on improving the efficiency of conventional methods. To specifically address the temporal precision of action detection, Gaidon et al. [4, 5] modeled the structure of action sequence with atomic action units (actoms). The explicit modeling of action units allows for matching more complete action unit sequences, rather than just partial content. However, this requires manual annotations for actoms, which can be subjective and burdensome. Our paper presented here aims to solve the same problem of precise temporal localization, but without requiring the difficult task of manual annotation of actoms.

Spatial-temporal localization. There have been active explorations about localizing action in space and time simultaneously. Jain et al. [10] and Soomro et al. [32] built their work on supervoxel. Recently, researchers treat this as a tracking problem [43, 8] by leveraging object detectors [11], especially human detectors [15, 18, 8, 45] to detect regions of interest in each frame and then output sequences of bounding boxes. Dense trajectories have also been exploited for extracting the action tubes [28, 37].

However, this problem is different from temporal localization, which is the main topic in this paper: (1) When using object detectors to find regions of interest, spatial-temporal localization assumes that the actions are performed by human or other pre-defined objects. (2) Spatial-temporal localization requires exhaustive annotations of objects of interest on every frames as training data. This makes it overwhelmingly time-consuming task in long untrimmed videos compared with the task of simply labeling the start and end time of an action depicted in the video, which is sufficient to satisfy many applications.

Object detection. Because our approach is encouraged by the success of deep learning approaches in object detection, we also review R-CNN and its variations. R-CNN consists of selective search, CNN feature extraction, SVM classification, and bounding box regression [7]. Fast R-CNN reshapes R-CNN into a single-stage using multi-task loss, and also has a RoI pooling layer to share the computation of one image in ConvNets [6]. Our work differs from R-CNN in the following aspects: (1) Temporal annotations in training videos can be diverse: some are cleanly trimmed action instances cut out from long videos, such as UCF101 [33], and some are untrimmed long videos but with temporal boundaries annotated for action instances, such as THU-MOS [14, 9]. We provide a paradigm about how to prepare training data for deep networks from such diverse annotations. (2) As proven in Faster R-CNN [29] which proposes region proposal networks, and DeepBox [20] which detects objectness to re-rank the results of R-CNN, using deep networks for learning objectness is effective and efficient. Therefore, we directly use deep networks to classify background and action to obtain candidate segments. (3) We remove the regression stage because learning regression for time shift and duration of video segment does not work well in practice, probably because actions can be quite diverse, and therefore does not contain consistent patterns for predicting start/end time. To achieve precise localization, we design the localization stage using new loss function to explicitly consider temporal overlap. This can decrease the score for the segment that has small overlap with the ground truth, and increase the segment of larger overlap. This also benefits post-processing steps, such as non-maximum suppression and top-k selection, to keep segment with higher temporal localization accuracy. Wan et al. [38] proposed a new loss function to resolve bottleneck in non-maximum suppression. However, all the bounding boxes within an image have to be considered at the same time, and this would require huge memory.

3. Detailed descriptions of Segment-CNNs

3.1. Problem setup

Problem definition. We denote a video as $X = \{x_t\}_{t=1}^T$, where $x_t$ is the $t$-th frame in $X$, and $T$ is the total number of frames in $X$. Each video $X$ is associated with a set of action temporal annotations $\Psi = \{(\psi_m, \psi'_m, k_m)\}_{m=1}^M$, where $M$ is the total number of action instances in $X$, and $k_m, \psi_m, \psi'_m$ are, respectively, action category of the instance $m$ and its starting time and ending time (measured by frame ID). $k_m \in \{1, \ldots, K\}$, where $K$ is the number of categories. During training, we have a set $\mathcal{T}$ of trimmed videos and a set $\mathcal{U}$ of untrimmed videos. Each trimmed video $X \in \mathcal{T}$ has $\psi_m = 1, \psi'_m = T$, and $M = 1$.

Multi-scale segment generation. First, each frame is resized to 171 (width) × 128 (height) pixels. For untrimmed video $X \in \mathcal{U}$, we conduct temporal sliding windows of varied lengths as 16, 32, 64, 128, 256, 512 frames with 75% overlap. For each window, we construct segment $s$ by uniformly sampling 16 frames. Consequently, for each untrimmed video $X$, we generate a set of candidates
\[ \Phi = \left\{ (s_h, \phi_h, \phi'_h) \right\}_{n=1}^{H} \] as input for the proposal stage, where \( H \) is the total number of sliding windows for \( X \), and \( \phi_m \) and \( \phi'_m \) are respectively starting time and ending time of the \( h \)-th segment \( s_h \). For trimmed video \( X \in \mathcal{T} \), we directly sample a segment \( s \) of 16 frames from \( X \) in uniform.

**Network architecture.** 3D ConvNets conducts 3D convolution/pooling which operates in spatial and temporal simultaneously, and therefore can capture both appearance and motion for action. Given the competitive performances on video classification tasks, our deep networks use 3D ConvNets as the basic architecture in all stages and follow the network architecture of [36]. All 3D pooling layers use max pooling and have kernel size of \( 2 \times 2 \) in spatial with stride \( 2 \), while vary in temporal. All 3D convolutional filters have kernel size \( 3 \) and stride \( 1 \) in all three dimensions. Using the notations \( conv \) (number of filters) for the 3D convolutional layer, \( pool \) (temporal kernel size, temporal stride) for the 3D pooling layer, and \( fc \) (number of filters) for the fully connected layer, the layout of these three types of layers in our architecture is as follows: \( conv1a(64) \) - \( pool1(1,1) \) - \( conv2a(128) \) - \( pool2(2,2) \) - \( conv3a(256) \) - \( conv3b(256) \) - \( pool3(2,2) \) - \( conv4a(512) \) - \( conv4b(512) \) - \( pool4(2,2) \) - \( conv5a(512) \) - \( conv5b(512) \) - \( pool5(2,2) \) - \( fc6(4096) \) - \( fc7(4096) \) - \( fc8(K + 1) \). Each input for this deep network is a segment \( s \) of dimension \( 171 \times 128 \times 16 \). C3D is training this network on Sports-1M train split [36], and we use C3D as the initialization for our networks in the proposal and classification stage.

### 3.2. Training procedure

#### The proposal stage

We train a CNNs model \( \Theta_{pro} \) in this stage as the background segment filter. Basically, \( fc8 \) has two nodes that correspondingly represent the background (rarely contains action of interest) and being-action (has significant portion belongs to the actions of interest).

We use the following strategy to construct training data \( S_{pro} = \{(s_n, k_n)\}_{n=1}^{N_{pro}} \), where label \( k_n \in \{0, 1\} \). For each segment of the trimmed video \( X \in \mathcal{T} \), we set its label as positive. For candidate segments from an untrimmed video \( X \in \mathcal{U} \) with temporal annotation \( \Psi \), we assign a label for each segment by evaluating its Intersection-over-Union (IoU) with each ground truth instance in \( \Psi : \text{if the highest IoU is larger than 0.7, we assign a positive label; (2) if the highest IoU is smaller than 0.3, we set it as the background. On the perspective of ground truth, if there is no segment that overlaps with a ground truth instance with IoU larger than 0.7, then we assign a positive label segment } s \text{ if } s \text{ has the largest IoU with this ground truth and its IoU is higher than 0.5. At last, we obtain } S_{pro} = \{(s_n, k_n)\}_{n=1}^{N_{pro}} \text{ which consists of all } N_T + N_U \text{ positive segments and } N_b \approx N_T + N_U \text{ randomly sampled background segments, where } N_{pro} = N_T + N_U + N_b.

In all experiments, we use a learning rate of 0.0001, with the exception of 0.01 for \( fc8 \), momentum of 0.9, weight decay factor of 0.0005, and drop the learning rate by a factor of 10 for every 10K iterations. The number of total iterations depends on the scale of dataset and will be clarified in Section 4.

Note that, compared with the multi-class background segments are removed in the proposal stage, this stage trains a classification CNNs model \( \Theta_{cls} \) for \( K \) action categories as well as background.

Preprocessing the training data \( S_{cls} \) follows a strategy similar to the proposal stage. Except when assigning label for positive segment, the classification stage explicitly indicates action category \( k_m \in \{1, \ldots, K\} \). Moreover, in order to balance the number of training data for each class, we reduce the number of background instances to \( N_b \approx \frac{N_T + N_U}{K} \).

As for parameters in SGD, the learning rate is 0.0001, with the exception of 0.01 for \( fc8 \), momentum is 0.9, weight decay factor is 0.0005, and the learning rate is divided by a factor of 2 for every 10K iterations, because the convergence shall be slower when the number of classes increases.

#### The classification stage

As illustrated in Figure 2, it is important to push up the prediction score of the segment with larger overlap with the ground truth instance and decrease the scores of the segment with smaller overlap, to make sure that the subsequent post-processing steps can choose segments with higher overlap over those with small overlap. Therefore, we propose this stage for training a classification network \( \Theta_{cls} \) with a new loss function, which takes IoU with ground truth instance into consideration.

Training data \( S_{loc} \) for the localization stage are augmented from \( S_{cls} \) by associating each segment \( s \) with the

![Figure 2. Typical case of bad localizations. Assume that the system outputs three predictions: A, B, C. Probably due to that there are some evidences during \( [t_1, t_2] \), and A has the highest prediction score. Therefore, the non-maximum suppression will keep A, remove B, and then keep C. However, actually we hope to remove A and C in non-maximum suppression, and keep B because B has the largest IoU with the ground truth instance.](image-url)
measurement of overlap, $v$. In specific, we set $v = 1$ for $s$ from trimmed video. If $s$ comes from untrimmed video and has positive label $k$, we set $v$ equal to the overlap (measured by IoU) of segment $s$ with the associated ground truth instance. If $s$ is a background segment, as we can see later, its overlap measurement $v$ will not affect our new loss function and gradient computation in back-propagation, and thus we simply set its $v$ as 1.

During each mini-batch, we have $N$ training samples $\{(s_n, k_n, v_n)\}_{n=1}^N$. For the $n$-th segment, the output vector of fc8 is $O_n$ and the prediction score vector after the softmax layer is $P_n$. Note that for the $i$-th class, $P_n^{(i)} = \frac{e^{O_n^{(i)}}}{\sum_{j=1}^{N} e^{O_n^{(j)}}}$. The new loss function is formed by combining $L_{\text{softmax}}$ and $L_{\text{overlap}}$:

$$L = L_{\text{softmax}} + \lambda \cdot L_{\text{overlap}},$$

where $\lambda$ balances the contribution from each part, and through empirical validation, we find that $\lambda = 1$ works well in practice. $L_{\text{softmax}}$ is the conventional softmax loss and is defined as

$$L_{\text{softmax}} = \frac{1}{N} \sum_n - \log \left( P_n^{(k_n)} \right),$$

which is effective for training deep networks for classification. $L_{\text{overlap}}$ is designed to jointly reduce the classification error and adjust the intensity of confidence score according to the extent of overlap:

$$L_{\text{overlap}} = \frac{1}{N} \sum_n \left( \frac{1}{2} \cdot \left( \frac{P_n^{(k_n)}}{v_n} - 1 \right) \cdot [k_n > 0] \right).$$

Here, $[k_n > 0]$ is equal to 1 when the true class label $k_n$ is positive, and it is equal to 0 when $k_n = 0$, which means the $s_n$ is a background training sample. $L_{\text{overlap}}$ is intended to boost the detection scores ($P$) of segments that have high overlaps ($v$) with ground truth instances, and reduce the scores of those with small overlaps. The hyper-parameter $\alpha$ controls the adjustment range for the intensity of the confidence score. The sensitivity of $\alpha$ is explored in Section 4.

In addition, the total gradient w.r.t output of the $i$-th node in fc8 is as follows:

$$\frac{\partial L}{\partial O_n^{(i)}} = \frac{\partial L_{\text{softmax}}}{\partial O_n^{(i)}} + \lambda \cdot \frac{\partial L_{\text{overlap}}}{\partial O_n^{(i)}},$$

in which

$$\frac{\partial L_{\text{softmax}}}{\partial O_n^{(i)}} = \left\{ \begin{array}{ll} \frac{1}{N} \cdot \left( P_n^{(k_n)} - 1 \right) & \text{if } i = k_n \\ \frac{1}{N} \cdot P_n^{(i)} & \text{if } i \neq k_n \end{array} \right.$$
ments with $P_{\text{pro}} \geq 0.7$. Then we evaluate the retained segments by $\Theta_{\text{loc}}$ to obtain action category predictions and confidence scores $P_{\text{loc}}$. During post-processing, we remove all segments predicted as the background and refine $P_{\text{loc}}$ by multiplying with class-specific frequency of occurrence for each window length in the training data to utilize window length distribution patterns. We apply non-maximum suppression using $P_{\text{loc}}$ with overlap threshold $\delta$ (measured by IoU). Finally, top-$\kappa$ selection is performed to select $\kappa$ segments according to $P_{\text{loc}}$.

4. Experiments

4.1. Datasets and setup

MEXaction2 [1]. This dataset contains two action classes: “BullChargeCape” and “HorseRiding”. This dataset consists of three subsets: INA videos, YouTube clips, and UCF101 Horse Riding clips. YouTube clips and UCF101 Horse Riding clips are trimmed, whereas INA videos are untrimmed and are approximately 77 hours in total. With regard to action instances with temporal annotation, they are divided into train set (1336 instances), validation set (310 instances), and test set (329 instances).

THUMOS 2014 [14]. The temporal action detection task in THUMOS Challenge 2014 is dedicated to localizing action instances in long untrimmed videos. The detection task involves 20 categories as indicated in Figure 4. The trimmed videos used for training are 2755 videos of these 20 actions in UCF101. The validation set contains 1010 untrimmed videos with temporal annotations of 3007 instances in total. The test set contains 3358 action instances from 1574 untrimmed videos, whereas only 213 of them contain action instances of interest. We exclude the remaining 1361 background videos in the test set.

4.2. Comparisons with state-of-the-art systems

Evaluation metrics. We follow the conventional metrics used in THUMOS Challenge to regard action temporal localization as a retrieval problem, and evaluate average precision (AP). A prediction is marked as correct only when it has the correct category prediction, and has IoU with ground truth instance larger than the overlap threshold (measured by IoU, and we use a threshold of 0.5 in all the following experiments). Note that redundant detections are not allowed.

Results on MEXaction2. We build our system based on Caffe [13] and C3D [36]. We only use the train set in MEXaction2 for training. The number of training iterations is 30K in the proposal stage, 20K in the classification stage, and 20K in the localization stage with $\alpha = 0.25$. We use IoU threshold $\delta = 0.4$ in non-maximum suppression and keep all non-background segments in top-$\kappa$ selection.

We denote our Segment-CNNs using the above settings as S-CNN and compare with typical dense trajectory features (DTF) with bag-of-visual-words representation. The results of DTF is provided by [1], which trains three SVM models with different set of negative samples and averages AP overall. According to Table 1, our Segment-CNNs achieves tremendous performance gain for “BullChargeCape” action and competitive performance for “HorseRiding” action. Figure 5 displays our prediction results for “BullChargeCape” and “HorseRiding”, respectively.

| Method        | mAP(%) |
|---------------|--------|
| DTF           | 0.3    |
| S-CNN         | 11.6   |

Table 1. Average precision on MEXaction2.

Results on THUMOS 2014: The instances in train set and validation set are used for training. The number of training iterations is 30K for all three stages. We again set $\alpha = 0.25$ in the localization stage, IoU threshold $\delta = 0.4$ in non-maximum suppression, and keep all non-background segments in top-$\kappa$ selection. We denote our Segment-CNNs using the above settings as S-CNN.

| Method          | mAP(%) |
|-----------------|--------|
| Karaman et al.  | 0.2    |
| Wang et al.     | 8.5    |
| Oneata et al.   | 15.0   |

S-CNN

Table 2. Mean average precision on THUMOS 2014.

As for comparisons, beyond DTF, several baseline systems incorporate frame-level deep networks and even utilize lots of other features: (1) Karaman et al. [16] used FV encoding of iDT with weighted saliency based pooling, and conducted late fusion with frame-level CNN features. (2) Wang et al. [41] built their system on iDT with FV representation and frame-level CNN features, and also performed post-processing to refine the detection results. (3) Oneata et al. [26] conducted localization using FV encoding of iDT on temporal sliding window, and performed post-processing following [24]. Finally, they conducted weighted fusion for the localization scores of temporal window and video-level scores generated by classifiers trained on iDT features, still

Note that the results reported in [1] use different evaluation metrics. To make them comparable, we re-evaluate their prediction results according to standard criteria mentioned in Section 4.2.

Note that the evaluation toolkit used in THUMOS 2014 has some bugs, and recently the organizers released a new toolkit with fair evaluation criteria. Here, we re-evaluate the submission results of all teams using the updated toolkit.
Efficiency analysis. Our approach is very efficient when compared with all other systems, which typically fuse different features, and therefore can become quite cumbersome. Most segments generated from sliding windows are removed in the first proposal stage, and thus the operations in classification and localization are greatly reduced. For each batch, the speed is around 1 second, and the average performance improves from 15.0% to 19.0%. We also show two prediction results for the THUMOS 2014 test set in Figure 6.

4.3. Designtation evaluation for each stage

The proposal stage. We compare S-CNN0 (removes the proposal stage completely) and two simplified networks used in the proposal stage: (1) S-CNN1 includes the proposal stage as described above (two nodes in fc8); (2) S-CNN2 further reduces the number of nodes in fc7 from 4096 to 512. Because of the smaller network architecture, S-CNN1 and S-CNN2 can reduce the number of operations for background segments, and therefore accelerate speed. In addition, the results listed in Table 4 demonstrate that keeping the proposal stage can also improve average precision because the model trained in the proposal is designed for filtering out background segments that lack action of interests. In addition, reducing networks scale in the proposal stage gradually degrades performance slightly.

Table 3. Average precision (%) for each class on THUMOS 2014 when \( \kappa = 20K \) during top-\( \kappa \) selection.

| Class ID | 7 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | mAP |
|----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S-CNN (w/o localization) | 11.7 | 19.5 | 8.4 | 23.6 | 24.4 | 14.1 | 10.2 | 16.0 | 10.1 | 16.1 | 16.6 | 15.1 | 3.4 | 39.2 | 29.2 | 11.9 | 10.5 | 10.5 | 14.4 | 20.1 | 5.2 | 16.6 |
| S-CNN (w/o classification) | 18.9 | 19.1 | 8.3 | 25.5 | 23.2 | 11.9 | 9.7 | 17.1 | 35.1 | 18.0 | 13.8 | 15.6 | 9.8 | 38.3 | 32.4 | 8.9 | 17.7 | 14.2 | 22.9 | 5.5 | 17.8 |
| S-CNN | 14.7 | 20.0 | 7.6 | 24.7 | 27.5 | 15.4 | 13.8 | 17.6 | 15.3 | 18.2 | 18.8 | 19.9 | 17.7 | 34.8 | 31.9 | 11.8 | 18.6 | 19.2 | 24.4 | 4.4 | 18.8 |

Table 4. mAP comparisons between no proposal stage and using the proposal stage of two different networks scales on THUMOS 2014.

| networks | S-CNN0 | S-CNN1 | S-CNN2 |
|----------|-------|-------|--------|
| mAP(%)   | 17.1  | 19.0  | 18.9   |

The classification stage. Although \( \Theta_{loc} \) can replace \( \Theta_{cls} \) during prediction, the classification stage is still important because fine-tuning on \( \Theta_{cls} \) results in better performance. To study the effects of the classification and localization stages individually, we compare three Segment-CNNs using different settings: (1) S-CNN: keep all three stages and settings in Section 4.2, and \( \Theta_{loc} \) is fine-tuned on \( \Theta_{cls} \); (2) S-CNN (w/o classification): remove the classification stage completely and have no \( \Theta_{cls} \) to serve as initialization for training \( \Theta_{loc} \); (3) S-CNN (w/o localization): remove the localization stage completely and use classification model \( \Theta_{cls} \) to produce predictions. As shown in Figure 7, S-CNN fine-tuned on \( \Theta_{cls} \) outperforms S-CNN (w/o classification) consistently when \( \kappa \) varies, and consequently the classification stage is necessary during training. Table 3 lists the average precision for each class when \( \kappa = 20K \) in top-\( \kappa \) selection.

The localization stage. Figure 7 and Table 3 also prove the effectiveness of the localization stage. By adding the localization stage, S-CNN can significantly improve performances compared with the baseline S-CNN (w/o localiza-
Figure 5. Prediction results for two action instances from MEXaction2 when the overlap threshold is set to 0.5 during evaluation. For each ground truth instance, we show two prediction results: A has the highest confidence score among the predictions associated with this ground truth, and B is an incorrect prediction. BullChargeCape: A is correct, but B is incorrect because each ground truth only allows one detection. HorseRiding: A is correct, but B is incorrect because each ground truth only allows one detection. The numbers shown with # are frame IDs.

Figure 6. Prediction results for two action instances from THUMOS 2014 test set when the overlap threshold is set to 0.5 during evaluation. For each ground truth instance, we show two prediction results: A has the highest confidence score among the predictions associated with this ground truth, and B is an incorrect prediction. ClearAndJerk: A is correct, but B is incorrect because its overlap IoU with ground truth is less than threshold 0.5. LongJump: A is correct, but B is incorrect because it has the wrong action category prediction - PoleVault.

Figure 7. Effects of the classification and localization stages. y-axis is mAP(%) on THUMOS 2014, and x-axis varies the depth $\kappa$ in top-$\kappa$ selection.

| $\alpha$ | 0.25 | 0.5 | 1   |
|---------|------|-----|-----|
| mAP(%)  | 24,444 | 23,229 | 23,050 |

Table 5. Performances on THUMOS 2014 as $\alpha$ varies. Here we keep all predictions during top-$\kappa$ selection, and therefore $\kappa$ is the total number of predictions.

5. Conclusion

We propose an effective multi-stage framework called Segment-CNNs to address action temporal localization in
untrimmed long videos. Through the above evaluation for each stage, we demonstrate the contribution from the proposal stage to identify candidate segments, the necessity of the classification stage to provide good initialization for training the localization model, and the effectiveness of the new loss function used in the localization stage to precisely localize action instances in time. In the future, we would like to extend our work to events and activities, which usually consist of multiple actions, therefore precisely localizing action instances in time can be helpful for their recognition and detection.

6. Acknowledgment

This work is supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20071. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI-NBC, or the U.S. Government. We thank Dong Liu and Guangnan Ye for the insightful suggestions.

References

[1] Mexaction2. http://mexculture.cnam.fr/xwiki/bin/view/Datasets/Mex+action+dataset, 2015. 6
[2] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. In ACM Computing Surveys, 2011. 1, 2
[3] G. Cheng, Y. Wan, A. N. Saudagar, K. Namuduri, and B. P. Buckles. Advances in human action recognition: A survey. 2015. 1, 2
[4] A. Gaidon, Z. Harchaoui, and C. Schmid. Actom sequence models for efficient action detection. In CVPR, 2011. 3
[5] A. Gaidon, Z. Harchaoui, and C. Schmid. Temporal localization of actions with actoms. In TPAMI, 2013. 3
[6] R. Girshick. Fast r-cnn. In ICCV, 2015. 1, 3
[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014. 1, 3
[8] G. Gkioxari and J. Malik. Finding action tubes. In CVPR, 2015. 3
[9] A. Gorban, H. Idrees, Y.-G. Jiang, A. R. Zamir, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. http://www.thumos.info/, 2015. 1, 2, 3
[10] M. Jain, J. van Gemert, H. Jégou, P. Bouthemy, and C. Snoek. Action localization with tubelets from motion. In CVPR, 2014. 3
[11] M. Jain, J. van Gemert, T. Mensink, and C. Snoek. Objects2action: Classifying and localizing actions without any video example. In ICCV, 2015. 3
[12] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. In TPAMI, 2013. 1, 3
[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In ACM MM, 2014. 6
[14] Y.-G. Jiang, J. Liu, A. R. 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. 1, 3, 6
[15] Z. Jiang, Z. Lin, and L. S. Davis. A unified tree-based framework for joint action localization, recognition and segmentation. In CVIU, 2013. 3
[16] S. Karaman, L. Seidenari, and A. D. Bimbo. Fast saliency based pooling of fischer encoded dense trajectories. 2014. 1, 6
[17] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F.-F. Li. Large-scale video classification with convolutional neural networks. In CVPR, 2014. 1, 3
[18] A. Kläser, M. Marszalek, C. Schmid, and A. Zisserman. Human focused action localization in video. In Trends and Topics in Computer Vision, 2012. 3
[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012. 3
[20] W. Kuo, B. Hariharan, and J. Malik. Deepbox: Learning objectness with convolutional networks. In ICCV, 2015. 1, 3
[21] K.-T. Lai, D. Liu, M.-S. Chen, and S.-F. Chang. Recognizing complex events in videos by learning key static-dynamic evidences. In ECCV, 2014. 2
[22] K.-T. Lai, F. X. Yu, M.-S. Chen, and S.-F. Chang. Video event detection by inferring temporal instance labels. In CVPR, 2014. 2
[23] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In CVPR, 2015. 1, 2
[24] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fischer vectors on a compact feature set. In ICCV, 2013. 1, 2, 6
[25] D. Oneata, J. Verbeek, and C. Schmid. Efficient action localization with approximately normalized fischer vectors. In CVPR, 2014. 3
[26] D. Oneata, J. Verbeek, and C. Schmid. The lear submission for joint action localization, recognition and segmentation. In TPAMI, 2014. 1, 3
[27] R. Poppe. A survey on vision-based human action recognition. In Image and vision computing, 2010. 1, 2
[28] M. M. Puscas, E. Sangineto, D. Culibrk, and N. Sebe. Unsupervised tube extraction using transductive learning and dense trajectories. In ICCV, 2015. 3
[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In NIPS, 2015. 1, 3
[30] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In NIPS, 2014. 1, 3
[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 3

[32] K. Soomro, H. Idrees, and M. Shah. Action localization in videos through context walk. In *ICCV*, 2015. 3

[33] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR-12-01*, 2012. 3

[34] A. Stoian, M. Ferecatu, J. Benois-Pineau, and M. Crucianu. Fast action localization in large scale video archives. In *TCSVT*, 2015. 3

[35] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *ACM MM*, 2015. 2

[36] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 1, 3, 4, 6

[37] J. van Gemert, M. Jain, E. Gati, and C. Snoek. Apt: Action localization proposals from dense trajectories. In *BMVC*, 2015. 3

[38] L. Wan, D. Eigen, and R. Fergus. End-to-end integration of a convolutional network, deformable parts model and non-maximum suppression. In *CVPR*, 2015. 3

[39] H. Wang, A. Klüser, C. Schmid, and C.-L. Liu. Action Recognition by Dense Trajectories. In *CVPR*, 2011. 1, 2

[40] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 1, 2

[41] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. 2014. 1, 6

[42] D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. In *Computer Vision and Image Understanding*, 2011. 1, 2

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

[44] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. In *CVPR*, 2015. 1, 3

[45] G. Yu and J. Yuan. Fast action proposals for human action detection and search. In *CVPR*, 2015. 3

[46] J. Yuan, Y. Pei, B. Ni, P. Moulin, and A. Kassim. Adsc submission at thumos challenge 2015. 2015. 1