CDC: Convolutional-De-Convolutional Networks for Precise Temporal Action Localization in Untrimmed Videos

Zheng Shou†, Jonathan Chan†, Alireza Zareian†, Kazuyuki Miyazawa‡, and Shih-Fu Chang†

†Columbia University, New York, NY, USA; ‡Mitsubishi Electric, Japan
{zs2262,jc4659,az2407,sc250}@columbia.edu, Miyazawa.Kazuyuki@cw.mitsubishielectric.co.jp

Abstract

Temporal action localization is an important yet challenging problem. Given a long, untrimmed video consisting of multiple action instances and complex background contents, we need not only to recognize their action categories, but also to localize the start time and end time of each instance. Many state-of-the-art systems use segment-level classifiers to select and rank proposal segments of predetermined boundaries. However, a desirable model should move beyond segment-level and make dense predictions at a fine granularity in time to determine precise temporal boundaries. To this end, we design a novel ***Convolutional-De-Convolutional*** *(CDC) network that places CDC filters on top of 3D ConvNets, which have been shown to be effective for abstracting action semantics but reduce the temporal length of the input data. The proposed CDC filter performs the required temporal upsampling and spatial downsampling operations simultaneously to predict actions at the frame-level granularity. It is unique in jointly modeling action semantics in space-time and fine-grained temporal dynamics. We train the CDC network in an end-to-end manner efficiently. Our model not only achieves superior performance in detecting actions in every frame, but also significantly boosts the precision of localizing temporal boundaries. Finally, the CDC network demonstrates a very high efficiency with the ability to process 500 frames per second on a single GPU server. Source code and trained models are available online at* https://bitbucket.org/columbiadvmm/cdc.

1. Introduction

Recently, temporal action localization has drawn considerable interest in the computer vision community [25, 15, 39, 66, 26, 68, 54, 47, 43, 74, 9, 18, 36]. This task involves two components: (1) determining whether a video contains specific actions (such as diving, jump, *etc*.) and (2) identifying temporal boundaries (start time and end time) of each action instance.

A typical framework used by many state-of-the-art systems [68, 54, 39, 66, 26] is fusing a large set of features and training classifiers that operate on sliding windows or segment proposals. Recently, an end-to-end deep learning framework called Segment-CNN (S-CNN) [47] based on 3D ConvNets [61] demonstrated superior performances both in efficiency and accuracy on standard benchmarks such as THUMOS'14 [25]. S-CNN consists of a proposal

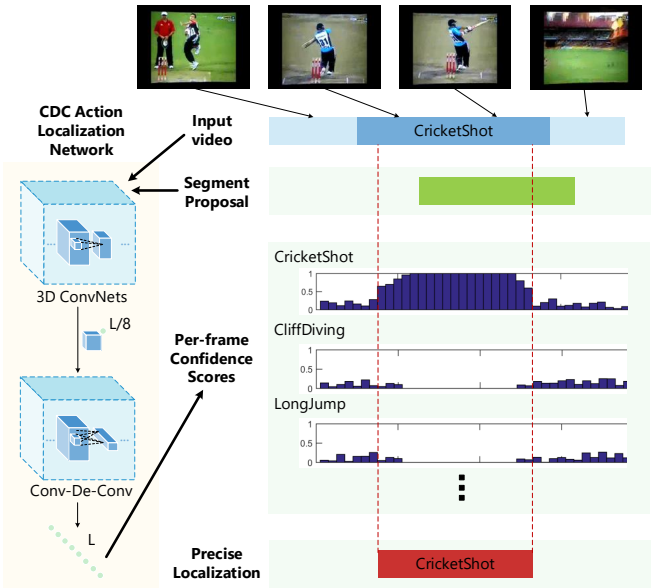

Figure 1. Our framework for precise temporal action localization. Given an input raw video, it is fed into our CDC localization network, which consists of 3D ConvNets for semantic abstraction and a novel CDC network for dense score prediction at the frame-level. Such fine-granular score sequences are combined with segment proposals to detect action instances with precise boundaries.

network for generating candidate video segments and a localization network for predicting segment-level scores of action classes. Although the localization network can be optimized to select segments with high overlaps with ground truth action instances, the detected action boundaries are still retained and thus are restricted to the pre-determined boundaries of a fixed set of proposal segments.

As illustrated in Figure 1, our goal is to refine temporal boundaries from proposal segments to precisely localize boundaries of action instances. This motivates us to move beyond existing practices based on segment-level predictions, and explicitly focus on the issue of fine-grained, dense predictions in time. To achieve this goal, some existing techniques can be adapted: (1) Single-frame classifiers operate on each frame individually; (2) Recurrent Neural Networks (RNN) further take into account temporal dependencies across frames. But both of them fail to explicitly model the spatio-temporal information in raw videos.

3D CNN [61, 47] has been shown that it can learn spatio-temporal abstraction of high-level semantics directly from raw videos but loses granularity in time, which is important for precise localization, as mentioned above. For example, layers from `conv1a` to `conv5b` in the well-known C3D architecture [61] reduce the temporal length of an input video by a factor of 8. In pixel-level semantic segmentation, deconvolution proves to be an effective upsampling method in both image [34, 45] and video [62] for producing output of the same resolution as the input. In our temporal localization problem, the temporal length of the output should be the same as the input video, but the spatial size should be reduced to 1x1. Therefore, we not only need to upsample in time but also need to downsample in space. To this end, we propose a novel **Convolutional-De-Convolutional** (CDC) filter, which performs convolution in space (for semantic abstraction) and de-convolution in time (for frame-level resolution) simultaneously. It is unique in jointly modeling the spatio-temporal interactions between summarizing high-level semantics in space and inferring fine-grained action dynamics in time. On top of 3D ConvNets, we stack multiple CDC layers to form our CDC network, which can achieve the aforementioned goal of temporal upsampling and spatial downsampling, and thereby can determine action categories and can refine boundaries of proposal segments to precisely localize action instances.

In summary, this paper makes three novel contributions:

(1) To the best of our knowledge, this is the first work to combine two reverse operations (*i.e.* convolution and deconvolution) into a joint CDC filter, which simultaneously conducts downsampling in space and upsampling in time to infer both high-level action semantics and temporal dynamics at a fine granularity in time.

(2) We build a CDC network using the proposed CDC filter to specifically address precise temporal action localization. The CDC network can be efficiently trained end-to-end from raw videos to produce dense scores that are used to predict action instances with precise boundaries.

(3) Our model outperforms state-of-the-art methods in video per-frame action labeling and significantly boosts the precision of temporal action localization over a wide range of detection thresholds.

2. Related work

Action recognition and detection. Early works mainly focus on simple actions in well-controlled environments and can be found in recent surveys [69, 41, 3]. Recently, researchers have started investigating untrimmed videos in the wild and have designed various features and techniques. We briefly review the following that are also useful in temporal action localization: frame-level Convolutional Neural Networks (CNN) trained on ImageNet [44] such as AlexNet [29], VGG [51], ResNet [16], *etc.*; 3D CNN architecture called C3D [61] trained on a large-scale sports video dataset [27] ; improved Dense Trajectory Feature (iDTF) [64, 65] consisting of HOG, HOF, MBH features extracted along dense trajectories with camera motion influences eliminated; key frame selection [13]; ConvNets adapted for using motion flow as input [50, 10, 67]; feature encoding with Fisher Vector (FV) [40, 38] and VLAD [23, 72].

There are also studies on spatio-temporal action detection, which aim to detect action regions with bounding boxes over consecutive frames. Various methods have been developed, from the perspective of supervoxel merging [20, 55, 56], tracking [70, 42, 63, 53], object detection and linking [28, 14, 76, 42, 63], spatio-temporal segmentation [31, 71], and leveraging still images [21, 59, 22].

Temporal action localization. Gaidon *et al.* [11, 12] introduced the problem of temporally localizing actions in untrimmed videos, focusing on limited actions such as "drinking and smoking" [30] and "open door and sit down" [8]. Later, researchers worked on building large-scale datasets consisting of complex action categories, such as THUMOS [25, 15] and MEXaction2 [57, 1, 58], and datasets focusing on fine-grained actions [35, 49, 48] or activities of high-level semantics [17]. The typical approach used in most systems [68, 54, 39, 66, 26] is extracting a pool of features, which are fed to train SVM classifiers, and then applying these classifiers on sliding windows or segment proposals for prediction. In order to design a model specific to temporal localization, Richard and Gall [43] proposed using statistical length and language modeling to represent temporal and contextual structures. Heilbron *et al.* [18] introduced a sparse learning framework for generating segment proposals of high recall.

Recently, deep learning methods showed improved performance in localizing action instances. RNN has been

widely used to model temporal state transitions over frames: Escorcia *et al*. [9] built a temporal action proposal system based on Long-Short Term Memory (LSTM); Yeung *et al*. [74] used REINFORCE to learn decision policies for a RNN-based agent; Yeung *et al*. [73] introduced MultiTHUMOS dataset of multi-label annotations for every frame in THUMOS videos and defined a LSTM network to model multiple input and output connections; Yuan *et al*. [77] proposed a pyramid of score distribution feature at the center of each sliding window to capture the motion information over multiple resolutions, and utilized RNN to improve inter-frame consistency; Sun *et al*. [60] leveraged web images to train LSTM model when only video-level annotations are available. In addition, Lea *et al*. [31] used temporal 1D convolution to capture scene changes when actions were being performed. Although RNN and temporal 1D convolution can model temporal dependencies among frames and make frame-level predictions, they are usually placed on top of deep ConvNets, which take a single frame as input, rather than directly modeling spatio-temporal characteristics in raw videos. Shou *et al*. [47] proposed an end-to-end Segment-based 3D CNN framework (S-CNN), which outperformed other RNN-based methods by capturing spatio-temporal information simultaneously. However, S-CNN lacks the capability to predict at a fine time resolution and to localize precise temporal boundaries of action instances.

De-convolution and semantic segmentation. Zeiler *et al*. [79] originally proposed de-convolutional networks for image decomposition, and later Zeiler and Fergus [78] repurposed de-convolutional filter to map CNN activations back to the input to visualize where the activations come from. Long *et al*. [34, 45] showed that deep learning based approaches can significantly boost performance in image semantic segmentation. They proposed Fully Convolutional Networks (FCN) to output feature maps of reduced dimensions, and then employed de-convolution for upsampling to make dense, pixel-level predictions. The fully convolutional architecture and learnable upsampling method are efficient and effective, and thus inspired many extensions [37, 19, 33, 4, 32, 80, 5, 6, 75].

Recently, Tran *et al*. [62] extended de-convolution from 2D to 3D and achieved competitive results on various voxel-level prediction tasks such as video semantic segmentation. This shows that de-convolution is also effective in the video domain and has the potential to be adapted for making dense predictions in time for our temporal action localization task. However, unlike the problem of semantic segmentation, we need to upsample in time but maintain downsampling in space. Instead of stacking a convolutional layer and a de-convolutional layer to conduct upsampling and downsampling separately, our proposed CDC filter learns a joint model to perform these two operations simultaneously, and proves to be more powerful and easier to train.

3. Convolutional-De-Convolutional networks

3.1. The need of downsampling and upsampling

C3D architecture, consisting of 3D ConvNets followed by three Fully Connected (FC) layers, has achieved promising results in video analysis tasks such as recognition [61] and localization [47]. Further, Tran *et al*. [62] experimentally demonstrated the 3D ConvNets, *i.e*. from `conv1a` to `conv5b`, to be effective in summarizing spatio-temporal patterns from raw videos into high-level semantics.

Therefore, we build our CDC network upon C3D. We adopt from `conv1a` to `conv5b` as the first part of our CDC network. For the rest of layers in C3D, we keep `pool5` to perform max pooling in height and width by a factor of 2 but retain the temporal length. Following conventional settings [61, 47, 62], we set the height and width of the CDC network input to 112x112. Given an input video segment of temporal length L, the output data shape of `pool5` is (512, $L/8$, 4, 4) [1]. Now in order to predict the action class scores at the original temporal resolution (frame-level), we need to upsample in time (from $L/8$ back to L), and downsample in space (from 4x4 to 1x1). To this end, we propose the CDC filter and design a CDC network to adapt the FC layers from C3D to perform the required upsample and downsample operations. Details are described in Sections 3.2 and 3.3.

3.2. CDC filter

In this section, we walk through a concrete example of adapting `FC6` layer in C3D to perform spatial downsampling by a factor of 4x4 and temporal upsampling by a factor of 2. For the sake of clarity, we focus on how a filter operates within one input channel and one output channel.

As explained in [34, 45], the FC layer is a special case of a convolutional layer (when the input data and the kernel have the same size and there is no striding and no padding). So we can transform `FC6` into `conv6`, which is shown in Figure 2 (a). Previously, a filter in `FC6` takes a 4x4 feature map from `pool5` as input and outputs a single value. Now, a filter in `conv6` can slide on $L/8$ feature maps of size 4x4 stacked in time and respectively output $L/8$ values in time. The kernel size of `conv6` is 4x4=16.

Although `conv6` performs spatial downsampling, the temporal length remains unchanged. To upsample in time, as shown in Figure 2 (b), a straightfoward solution adds a de-convolutional layer `deconv6` after `conv6` to double the temporal length while maintaining the spatial size. The kernel size of `deconv6` is 2. Therefore, the total number of pa-

[1] We denote the shape of data in the networks using the form of (number of channels, temporal length, height, width) and the size of feature map, kernel, stride, zero padding using (temporal length, height, width).

rameters for this solution (separated conv6 and deconv6) is 4x4+2=18.

However, this solution conducts temporal upsampling and spatial downsampling in a separate manner. Instead, we propose the CDC filter CDC6 to jointly perform these two operations. As illustrated in Figure 2 (c), a CDC6 filter consists of two independent convolutional filters (the red one and the green one) operating on the same input 4x4 feature map. Each of these convolutional filters has the same kernel size as the filter in conv6 and separately outputs one single value. So each 4x4 feature map results in 2 outputs in time. As the CDC filter slides on $L/8$ feature maps of size 4x4 stacked in time, this input feature volume of temporal length $L/8$ is upsampled in time to $L/4$, and its spatial size is reduced to 1x1. Consequently, in space this CDC filter is equivalent to a 2D convolutional filter of kernel size 4x4; in time it has the same effect as a 1D de-convolutional filter of kernel size 2, stride 2, padding 0. The kernel size of such a joint filter in CDC6 is 2x4x4=32, which is larger than the separate convolution and de-convolution solution (18).

Therefore, a CDC filter is more powerful for jointly modeling high-level semantics and temporal dynamics: each output in time comes from an independent convolutional kernel dedicated to this output (the red/green node corresponds to the red/green kernel); however, in the separate convolution and de-convolution solution, different outputs in time share the same high-level semantics (the blue node) outputted by one single convolutional kernel (the blue one).

Having more parameters makes the CDC filter harder to learn. To remedy this issue, we propose a method to adapt the pre-trained FC6 layer in C3D to initialize CDC6. After we convert FC6 to conv6, conv6 and CDC6 have the same number of channels (*i.e.* 4,096) and thus the same number of filters. Each filter in conv6 can be used to initialize its corresponding filter in CDC6: the filter in conv6 (the blue one) has the same kernel size as each of these two convolutional filters (the red one and the green one) in the CDC6 filter and thus can serve as the initialization for them both.

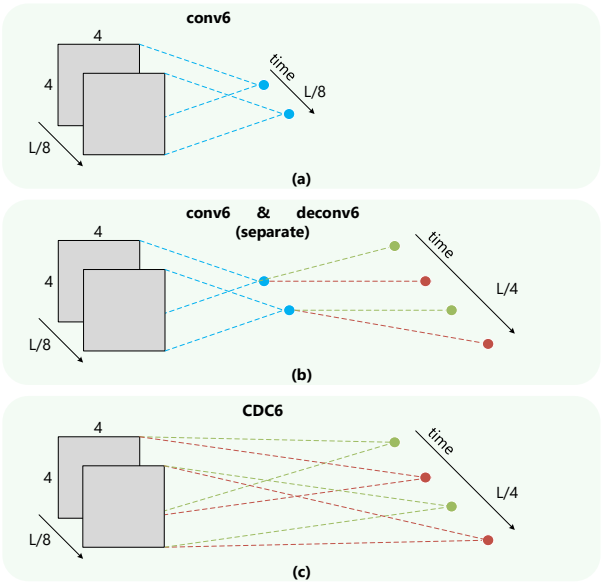

Figure 2. Illustration of how a filter in conv6, deconv6, CDC6 operates on pool5 output feature maps (grey rectangles) stacked in time. In each panel, dashed lines with the same color indicate the same filter sliding over time. Nodes stand for outputs.

Generally, assume that a CDC filter F of kernel size (k_l, k_h, k_w) takes the input receptive field X of height k_h and width k_w, and produces Y that consists of k_l successive outputs in time. For the example given in Figure 2 (c), we have $k_l = 2$, $k_h = 4$, $k_w = 4$. Given the indices $a \in \{1, ..., k_h\}$ and $b \in \{1, ..., k_w\}$ in height and width respectively for X and the index $c \in \{1, ..., k_l\}$ in time for Y: during the forward pass, we can compute Y by

$$Y\left[c\right] = \sum_{a=1}^{k_h} \sum_{b=1}^{k_w} F\left[c, a, b\right] \cdot X\left[a, b\right]; \tag{1}$$

during the back-propagation, our CDC filter follows the chain rule and propagates gradients from Y to X via

$$X\left[a, b\right] = \sum_{c=1}^{k_l} F\left[c, a, b\right] \cdot Y\left[c\right]. \tag{2}$$

A CDC filter F can be regarded as coupling a series of convolutional filters (each one has kernel size k_h in height and k_w in width) in time with a shared input receptive field X, and at the same time, F performs 1D de-convolution with kernel size k_l in time. In addition, the cross-channel mechanisms within a CDC layer and the way of adding biases to the outputs of the CDC filters follow the conventional strategies used in convolutional and de-convolutional layers.

3.3. Design of CDC network architecture

In Figure 3, we illustrate our CDC network for labeling every frame of a video. The final output shape of the CDC network is (K+1, L, 1, 1), where K+1 stands for K action categories plus the background class. As described in Section 3.1, from conv1a to pool5, the temporal length of an input segment has been reduced from L to $L/8$. On top of pool5, in order to make per-frame predictions, we adapt FC layers in C3D as CDC layers to perform temporal upsampling and spatial downsampling operations. Following previous de-convolution works [62, 34, 45], we upsample in time by a factor of 2 in each CDC layer, to gradually increase temporal length from $L/8$ back to L.

In the previous Section 3.2, we provide an example of how to adapt FC6 as CDC6, performing temporal 1D de-convolution of kernel size 2, stride 2, padding 0. For CDC6 in the CDC network, we construct a CDC filter with 4 convolutional filters instead of 2, and thus its temporal kernel

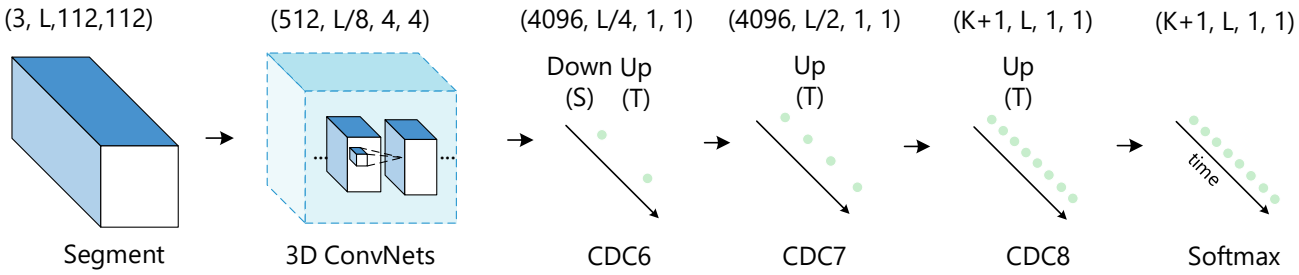

Figure 3. Architecture of a typical CDC network. Following the notations indicated in the footnote 1, the top row lists the shape of output data at each layer. (1) A video segment is first fed into 3D ConvNets and the temporal length reduces from L to $L/8$. (2) `CDC6` has kernel size (4, 4, 4), stride (2, 1, 1), padding (1, 0, 0), and therefore reduces both height and width to 1 while increases the temporal length from $L/8$ to $L/4$. Both `CDC7` and `CDC8` have kernel size (4, 1, 1), stride (2, 1, 1), padding (1, 0, 0), and hence both `CDC7` and `CDC8` further perform upsampling in time by a factor of 2, and thus the temporal length is back to L. (3) A frame-wise softmax layer is added on top of `CDC8` to obtain confidence scores for every frame. Each channel stands for one class.

size in time increases from 2 to 4. We set the corresponding stride to 2 and padding to 1. Now each 4x4 feature map produces 4 output nodes, and every two consecutive feature maps have 2 nodes overlapping in time. Consequently, the temporal length of input is still upsampled by `CDC6` from $L/8$ to $L/4$, but each output node sums contributions from two consecutive input feature maps, allowing temporal dynamics in input to be taken into account.

Likewise, we can adapt `FC7` as `CDC7`, as indicated in Figure 3. Additionally, we retain the Relu layers and the Dropout layers with 0.5 dropout ratio from C3D to attach to both `CDC6` and `CDC7`. `CDC8` corresponds to `FC8` but cannot be directly adapted from `FC8` because the classes in `FC8` and `CDC8` are different. Since each channel stands for one class, `CDC8` has K+1 channels. Finally, the `CDC8` output is fed into a frame-wise softmax layer `Softmax` to produce per-frame scores. During each mini-batch with N training segments, for the n-th segment, the `CDC8` output O_n has the shape (K+1, L, 1 ,1). For each frame, performing the conventional softmax operation and computing the softmax loss and gradient are independent of other frames. Corresponding to the t-th frame, the `CDC8` output $O_n\,[t]$ and `Softmax` output $P_n\,[t]$ both are vectors of K+1 values. Note that for the i-th class, $P_n^{(i)}\,[t] = \frac{e^{O_n^{(i)}[t]}}{\sum_{j=1}^{K+1} e^{O_n^{(j)}[t]}}$. The total loss $\mathcal{L}$ is defined as:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{L}\left(-\log\left(P_n^{(z_n)}\,[t]\right)\right), \tag{3}$$

where z_n stands for the ground truth class label for the n-th segment. The total gradient w.r.t the output of i-th channel/class and t-th frame in `CDC8` is the summation over all N training segments of:

$$\frac{\partial \mathcal{L}}{\partial O_n^{(i)}\,[t]} = \begin{cases} \frac{1}{N}\cdot\left(P_n^{(z_n)}\,[t]-1\right) & \text{if } i = z_n \\ \frac{1}{N}\cdot P_n^{(i)}\,[t] & \text{if } i \neq z_n \end{cases}. \tag{4}$$

3.4. Training and prediction

Training data construction. In theory, because both the convolutional filter and the CDC filter slide over the input, they can be applied to input of arbitrary size. Therefore, our CDC network can operate on videos of variable lengths. Due to GPU memory limitations, in practice we slide a temporal window of 32 frames without overlap on the video and feed each window individually into the CDC network to obtain dense predictions in time. From the temporal boundary annotations, we know the label of every frame. Frames in the same window can have different labels. To prevent including too many background frames for training, we only keep windows that have at least one frame belonging to actions. Therefore, given a set of training videos, we obtain a training collection of windows with frame-level labels.

Optimization. We use stochastic gradient descent to train the CDC network with the aforementioned frame-wise softmax loss. Our implementation is based on Caffe [24] and C3D [61]. The learning rate is set to 0.00001 for all layers except for `CDC8` layer where the learning rate is 0.0001 since `CDC8` is randomly initialized. Following conventional settings [61, 47], we set momentum to 0.9 and weight decay to 0.005.

C3D [61] is trained on Sports-1M [27] and can be used to directly initialize `conv1a` to `conv5b`. `CDC6` and `CDC7` are initialized by `FC6` and `FC7` respectively using the strategy described in the Section 3.2. In addition, since `FC8` in C3D and `CDC8` in the CDC network have the different number of channels, we randomly initialize `CDC8`. With such initialization, our CDC network turns out to be very easy to train and converges quickly, *i.e.* 4 training epochs (within half a day) on THUMOS'14 .

Fine-grained prediction and precise localization. During testing, after applying the CDC network on the whole video,

we can make predictions for every frame of the video. Through thresholding on confidence scores and grouping adjacent frames of the same label, it is possible to cut the video into segments and produce localization results. But this method is not robust to noise, and designing temporal smoothing strategies turns out to be ad hoc and non-trivial. Recently, researchers developed some efficient segment proposal methods [47, 9] to generate a small set of candidate segments of high recall. Utilizing these proposals for our localization model not only bypasses the challenge of grouping adjacent frames, but also achieves considerable speedup during testing, because we only need to apply the CDC network on the proposal segments instead of the whole video.

Since these proposal segments only have coarse boundaries, we propose using fine-grained predictions from the CDC network to localize precise boundaries. First, to look at a wider interval, we extend each proposal segment's boundaries on both sides by the percentage α of the original segment length. We set α to 1/8 for all experiments. Then, similar to preparing training segments, we slide temporal windows without overlap on the test videos. We only need to keep test windows that overlap with at least one extended proposal segment. We feed these windows into our CDC network and generate per-frame action classes scores.

The category of each proposal segment is set to the class with the maximum average confidence score over all frames in the segment. If a proposal segment does not belong to the background class, we keep it and further refine its boundaries. Given the score sequence of the predicted class in the segment, we perform Gaussian kernel density estimation and obtain its mean μ and standard deviation σ. Starting from the boundary frame at each side of the extended segment and moving towards its middle, we shrink its temporal boundaries until we reach a frame with the confidence score no lower than μ - σ. Finally, we set the prediction score of the segment to the average confidence score of the predicted class over frames in the refined segment of boundaries.

methods	mAP
Single-frame CNN [51]	34.7
Two-stream CNN [50]	36.2
LSTM [7]	39.3
MultiLSTM [73]	41.3
C3D + LinearInterp	37.0
Conv & De-conv	41.7
CDC (fix 3D ConvNets)	37.4
CDC	**44.4**

Table 1. Per-frame labeling mAP on THUMOS'14 .

4. Experiments

4.1. Per-frame labeling

We first demonstrate the effectiveness of our model in predicting accurate labels for every frame. Note that this task can accept an input of multiple frames to take into account temporal information. We denote our model as **CDC**.

THUMOS'14 [25]. The temporal action localization task in THUMOS Challenge 2014 involves 20 actions. We use 2,755 trimmed training videos and 1,010 untrimmed validation videos (3,007 action instances) to train our model. For testing, we use all 213 test videos (3,358 action instances) which are not entirely background videos.

Evaluation metrics. Following conventional metrics [73], we treat the per-frame labeling task as a retrieval problem. For each action class, we rank all frames in the test set by their confidence scores for that class and compute Average Precision (AP). Then we average over all classes to obtain mean AP (mAP).

Comparisons. In Table 1, we first compare our CDC network (denoted by CDC) with some state-of-the-art models (results are quoted from [73]): (1) Single-frame CNN: the frame-level 16-layer VGG CNN model [51]; (2) Two-stream CNN: the frame-level two-stream CNN model proposed in [50], which has one stream for pixel and one stream for optical flow; (3) LSTM: the basic per-frame labeling LSTM model of 512 hidden units [7] on the top of VGG CNN `FC7` layer; (4) MultiLSTM: a LSTM model developed by Yeung *et al*. [73] to process multiple input frames together with temporal attention mechanism and output predictions for multiple frames. Single-frame CNN only takes into account appearance information. Two-stream CNN models appearance and motion information separately. LSTM based models can capture temporal dependencies across frames but do not model motion explicitly. Our **CDC** model is based on 3D convolutional layers and CDC layers, which can operate on spatial and temporal dimensions simultaneously, achieving the best performance.

In addition, we compare CDC with other C3D based approaches that use different upsampling methods. (1) C3D + LinearInterp: we train a segment-level C3D using the same set of training segments whose segment-level labels are determined by the majority vote. During testing we perform linear interpolation to upsample segment-level predictions as frame-level. (2) Conv & De-conv: `CDC7` and `CDC8` in our CDC network keep the spatial data shape unchanged and therefore can be also regarded as de-convolutional layers. For `CDC6`, we replace it with a convolutional layer `conv6` and a separate de-convolutional layer `deconv6` as shown in

IoU threshold	0.3	0.4	0.5	0.6	0.7
Karaman *et al*. [26]	0.5	0.3	0.2	0.2	0.1
Wang *et al*. [66]	14.6	12.1	8.5	4.7	1.5
Heilbron *et al*. [18]	-	-	13.5	-	-
Escorcia *et al*. [9]	-	-	13.9	-	-
Oneata *et al*. [39]	28.8	21.8	15.0	8.5	3.2
Richard and Gall [43]	30.0	23.2	15.2	-	-
Yeung *et al*. [74]	36.0	26.4	17.1	-	-
Yuan *et al*. [77]	33.6	26.1	18.8	-	-
S-CNN [47]	36.3	28.7	19.0	10.3	5.3
C3D + LinearInterp	36.0	26.4	19.6	11.1	6.6
Conv & De-conv	38.6	28.2	22.4	12.0	7.5
CDC (fix 3D ConvNets)	36.9	26.2	20.4	11.3	6.8
CDC	**40.1**	**29.4**	**23.3**	**13.1**	**7.9**

Table 2. Temporal action localization mAP on THUMOS'14 as the overlap IoU threshold used in evaluation varies from 0.3 to 0.7. - indicates that results are unavailable in the corresponding papers.

Figure 2 (b). The CDC model outperforms these baselines because the CDC filter can simultaneously model high-level semantics and temporal action dynamics. We also evaluate the CDC network with fixed weights in 3D ConvNets and only fine-tune CDC layers, resulting in a minor performance drop. This implies that it is helpful to train CDC networks in an end-to-end manner so that the 3D ConvNets part can be trained to summarize more discriminative information for CDC layers to infer more accurate temporal dynamics.

4.2. Temporal action localization

Given per-frame labeling results from the CDC network, we generate proposals, determine class category, and predict precise boundaries following Section 3.4. Our approach is applicable to any segment proposal method. Here we conduct experiments on THUMOS'14, and thus employ the publicly available proposals generated by the S-CNN proposal network [47], which achieves high recall on THUMOS'14 . Finally, we follow [73, 47] to perform standard post-processing steps such as non-maximum suppression.

Evaluation metrics. Localization performance is also evaluated by mAP. Each item in the rank list is a predicted segment. The prediction is correct when it has the correct category and its temporal overlap IoU with the ground truth is larger than the threshold. Redundant detections for the same ground truth instance are not allowed.

Comparisons. As shown in Table 2, **CDC** achieves much better results than all the other state-of-the-art methods, which have been reviewed in Section 2. Compared to the proposed CDC model: the typical approach of extracting a set of features to train SVM classifiers and then applying the trained classifiers on sliding windows or segment proposals (Karaman *et al*. [26], Wang *et al*. [66], Oneata *et al*. [39], Escorcia *et al*. [9]) does not directly address the temporal localization problem. Systems encoding iDTF with FV (Heilbron *et al*. [18], Richard and Gall [43]) cannot learn spatio-temporal patterns directly from raw videos to make predictions. RNN/LSTM based methods (Yeung *et al*. [74], Yuan *et al*. [77]) are unable to explicitly capture motion information beyond temporal dependencies. S-CNN can effectively capture spatio-temporal patterns from raw videos but lacks the ability of adjusting boundaries from proposal candidates. With the proposed CDC filter, the CDC network can determine confidence scores at a fine granularity, beyond segment-level prediction, and hence precisely localize temporal boundaries. In addition, we employ per-frame predictions of other methods indicated in Table 1 (C3D + LinearInterp, Conv & De-conv, CDC with fixed 3D ConvNets) to perform temporal localization based on S-CNN proposal segments. As shown in Table 2, the performance of the CDC network is still better, because more accurate predictions at the same temporal granularity can be used to predict more accurate label and more precise boundaries for the same input proposal segment. In Figure 4, we illustrate how our model refines boundaries from segment proposal to precisely localize action instance in time.

4.3. Discussions

The necessity of predicting at a fine granularity in time. In Figure 5, we compare CDC networks predicting action scores at different temporal granularities. When the temporal granularity increases, mAP increases accordingly. This demonstrates the importance of predicting at a fine-granularity for achieving precise localization.

Efficiency analysis. The CDC network is compact and demands little storage, because it can be trained from raw videos directly to make fine-grained predictions in an end-to-end manner without the need to cache intermediate features. A typical CDC network such as the example in Figure 3 only requires around 1GB storage.

Our approach is also fast. Compared with segment-level prediction methods such as S-CNN localization network [47], CDC has to perform more operations due to the need of making predictions at every frame. Therefore, when the proposal segment is long, CDC is less efficient for the sake of achieving more accurate boundaries. But in the case of short proposal segments, since these proposals usually are densely overlapped, segment-level methods have to process a large number of segments one by one. However, CDC networks only need to process each frame once, and thus it can avoid redundant computations. On a NVIDIA Titan X GPU of 12GB memory, the speed of a CDC network is around 500 Frames Per Second (FPS), which means it can

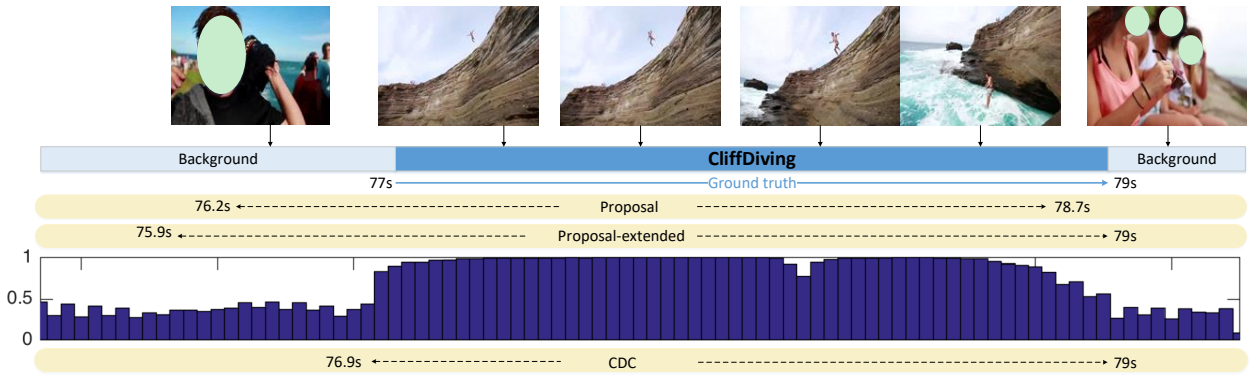

Figure 4. Visualization of the process of refining temporal boundaries for a proposal segment. Horizontal axis stands for time. From the top to the bottom: (1) frame-level ground truths for a CliffDiving instance in an input video with some representative frames; (2) a corresponding proposal segment; (3) the proposal segment after extension; (4) the per-frame score of detecting CliffDiving predicted by the CDC network; (5) the predicted action instance after the refinement using CDC.

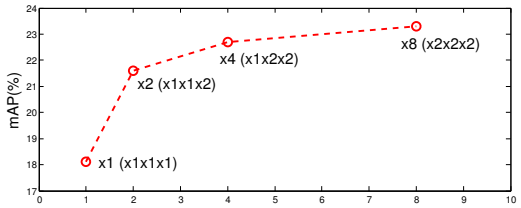

Figure 5. mAP gradually increases when the temporal granularity of CDC network prediction increases from x1 (one label for every 8 frames) to x8 (one label per frame). Each point corresponds to **x total upscaling factor (x `CDC6` upscaling factor x `CDC7` upscaling factor x `CDC8` upscaling factor)** in time. We conduct the evaluation on THUMOS'14 with IoU 0.5.

process a 20s long video clip of 25 FPS within one second.

Temporal activity localization. Furthermore, we found that our approach is also useful for localizing activities of high-level semantics and complex components. We conduct experiments on ActivityNet Challenge 2016 dataset [17, 2], which involves 200 activities, and contains around 10K training videos (15K instances) and 5K validation videos (7.6K instances). Each video has an average of 1.65 instances with temporal annotations. We train on the training videos and test on the validation videos. Since no activity proposal results of high quality exist, we apply the trained CDC network to the results of the first place winner [68] in this Challenge to localize more precise boundaries. As shown in Table 5, they achieve high mAP when the IoU in evaluation is set to 0.5, but mAP drops rapidly when the evaluation IoU increases. After using the per-frame predictions of our CDC network to refine temporal boundaries of their predicted segments, we gain significant improvements particularly when the evaluation IoU is high (*i.e.* 0.75). This means that after the refinement, these segments have more precise boundaries and have larger overlap with ground truth instances.

mAP	0.5	0.75	0.95	Average-mAP
before	45.1	4.1	0.0	16.4
after	45.3	26.0	0.2	23.8

Table 3. Temporal localization mAP on ActivityNet Challenge 2016 [2] of Wang and Tao [68] before and after the refinement step using our CDC network. We follow the official metrics used in [2] to evaluate the average mAP.

5. Conclusion and future works

In this paper, we propose a novel CDC filter to simultaneously perform spatial downsampling (for spatio-temporal semantic abstraction) and temporal upsampling (for precise temporal localization), and design a CDC network to predict actions at frame-level. Our model significantly outperforms all other methods both in the per-frame labeling task and the temporal action localization task. Supplementary descriptions of the implementation details and additional experimental results are available in [46].

6. Acknowledgment

The project was supported by Mitsubishi Electric, and also by Award No. 2015-R2-CX-K025, awarded by the National Institute of Justice, Office of Justice Programs, U.S. Department of Justice. The opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect those of the Department of Justice. The Tesla K40 used for this research was donated by the NVIDIA Corporation. We thank Wei Family Private Foundation for their support for Zheng Shou, and anonymous reviewers for their valuable

comments.

7. Appendix

7.1. Additional justification of the motivation

As mentioned in the paper, the traditional approaches use segment-level detection, in which segment proposals are analyzed to predict the action class in each segment. Such approaches are limited by the fixed segment lengths and boundary locations, and thus inadequate for finding precise action boundaries. Here we proposed a novel model to first predict actions at fine-level and then use such fine-grained score sequences to accurately detect the action boundaries. The fine-grained score sequence also offers natural ways to determine the score threshold needed in refining boundaries at the frame level. Also, though not emphasized in the paper, the fine-level score sequence can also be used to select precise keyframes or discover sub-actions within an action.

Following the reviewer's suggestion, we also computed the frame-to-frame score gradient using the frame-level detection results. As shown in Figure 6, the frame-level gradient peaks nicely correlate with the action boundaries, confirming the intuition of using the fine-level detection results. Also, as shown in Figure 5 in the paper, when the temporal granularity increases, localization performance increases accordingly. Finally, our motivation is quantitatively justified by the good results on two standard benchmarks as shown in Section 4.

7.2. Additional implementation details

Temporal boundary refinement. Here, we provide details and pseudo-codes for temporal boundary refinement presented in Section 3.4. Algorithm 1 is used to refine boundaries of each proposal segment. Also, our source codes can be found at https://bitbucket.org/columbiadvmm/cdc.

Algorithm 1 Temporal Boundary Refinement

Input: A proposal segment of starting frame index t_s and ending frame index t_e, the percentage parameter of segment length expansion α, the first frame index v_s and the last frame index v_e of the video containing the proposal segment, the total number of categories K

Output: the refined starting frame index ${t_s}'$ and ending frame index ${t_e}'$, the predicted category c, the predicted confidence score s

// Extend boundaries on both sides by the percentage of the original segment length
${t_s}' = \max\left(v_s, t_s - \alpha \cdot (t_e - t_s + 1)\right)$
${t_e}' = \min\left(v_e, t_e + \alpha \cdot (t_e - t_s + 1)\right)$
// Feed frames into the CDC network to produce the confidence score matrix $\mathbf{P} \in \Re^{({t_e}-{t_s}+1)\times K}$
$\mathbf{P}$ = **CDC**(frames from ${t_s}'$ to ${t_e}'$)
assign c as the category with the maximum average confidence score over all frames from ${t_s}'$ to ${t_e}'$
// Estimate the mean μ and the standard deviation σ
μ, σ = Gaussian Kernel Density Estimation($\mathbf{P}\,[:, c]$)
$\beta = \mu - \sigma$ // Compute the score threshold
// Refine the starting time
for $i_s = 1, 2, \ldots, ({t_e}' - {t_s}' + 1)$ **do**
  if $\mathbf{P}\,[i_s, c] >= \beta$ **then**
    break
  end if
end for
// Refine the ending time
for $i_e = ({t_e}' - {t_s}' + 1), \ldots, 2, 1$ **do**
  if $\mathbf{P}\,[i_e, c] >= \beta$ **then**
    break
  end if
end for
${t_e}' = {t_s}' + i_e - 1$
${t_s}' = {t_s}' + i_s - 1$
$s = \frac{\sum_{i=i_s}^{i_e} \mathbf{P}[i,c]}{t_e - t_s + 1}$ // Compute the average score
return ${t_s}'$, ${t_e}'$, c, s

Discussions about the window length used during creating mini-batches. During mini-batch construction, ideally we would like to set the window length as longer as possible. Therefore, when CDC processes each window, it can take into account more temporal contextual information. However, due to the limitation of the GPU memory, if the window length is too high, we have to set the number of training samples for each mini-batch to be very small, which will make the optimization unstable and thus the training procedure cannot converge well. Also, a long window usually contains much more background frames than action frames and thus we need to further handle the data imbalance issue. During experiments, we conduct a

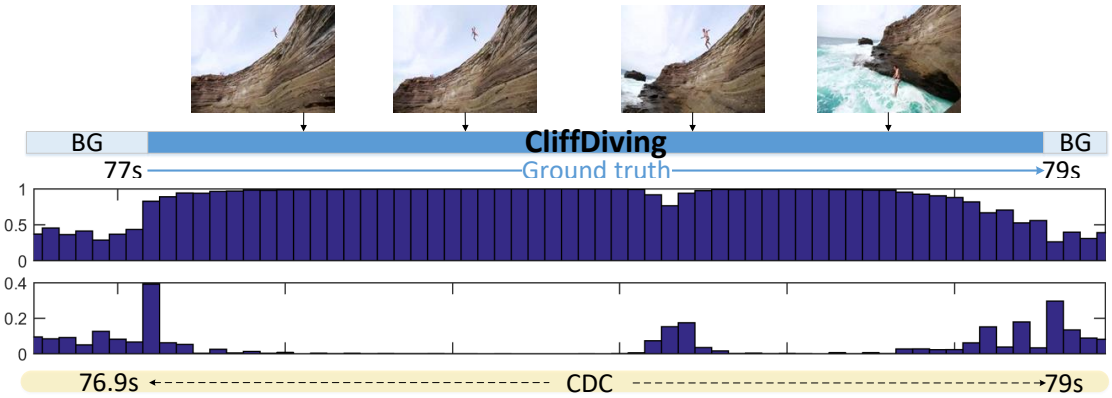

Figure 6. We use the frame-level detection scores (3rd row) to compute the absolute frame-to-frame score differences (4th row), which show high correlations with the true action boundaries.

grid search of window length in 16, 32, 64, 128, 256, 512 and empirically found that setting the window length to 32 frames is a good trade-off on a single NVIDIA Titan X GPU of 12GB memory: (1) we can include sufficient temporal contextual information to achieve good accuracy and (2) we can set the batch size as 8 to guarantee stable optimization.

7.3. Additional experiments

Sensitivity analysis. When we extend the segment proposal, the percentage α of the original proposal length should not be too small so that our model can consider a wider interval and not be too large to include too many irrelevant frames. As shown in Table 4, the system has stable performances when α varies within a reasonable range.

α	1/8	1/7	1/6	1/5	1/4
mAP	23.3	23.2	23.1	23.1	23.6

Table 4. mAP on THUMOS'14 with the evaluation IoU set to 0.5 when we vary the extension percentage α of the original proposal length from 1/8 to 1/4.

Additional results on ActivityNet. We expand the comparisons on ActivityNet validation set to include results provided by additional top performers [51, 52] in ActivityNet Challenge 2016. As shown in Table 5, our method CDC outperforms all other methods. As shown in Table 6, CDC also performs the best on ActivityNet test set.

Discussions about other proposal methods. As shown in Table 7, we evaluate temporal localization performances of CDC based on other proposals on THUMOS'14.

On ActivityNet, the proposals currently used in Section 4 from [68] is a reasonable choice - its recall is 0.681 with 56K proposals when evaluate at IoU=0.5 on the validation set. We also have considered using other state-of-the-art proposal methods: (1) The ActivityNet challenge provides proposals computed by [18], but it has a low recall at 0.527 on the validation set with 441K proposals, which contain a lot of false alarms. (2) DAPs [9] advocates that train proposal model on THUMOS and then generalize the model to ActivityNet. Due to lack training data from ActivityNet, DAPs has a quite low recall at around 0.23 and is not a reasonable proposal candidate. (3) S-CNN [47] is designed for instance-level detection. However, ground truth annotations in ActivityNet do not distinguish consecutive instances - one ground truth interval can contain multiple activity instances. Also, for activities of high-level semantics, it is ambiguous to define what is an individual activity instance.

IoU threshold	0.5	0.75	0.95	Ave-mAP
Singh and Cuzzolin [54]	22.7	10.8	0.3	11.3
Singh [52]	26.0	15.2	2.6	14.6
Wang and Tao [68]	45.1	4.1	0.0	16.4
CDC	45.3	26.0	0.2	23.8

Table 5. Additional baseline results of temporal localization mAP on ActivityNet Challenge 2016 [2] validation set. The baseline results are kindly provided by the authors of [54, 52, 68].

IoU threshold	0.5	0.75	0.95	Ave-mAP
Singh and Cuzzolin [54]	36.4	11.1	0.1	17.8
Singh [52]	28.7	17.8	2.9	17.7
Wang and Tao [68]	42.5	2.9	0.1	14.6
CDC (train)	43.1	25.6	0.2	22.9
CDC (train+val)	43.0	25.7	0.2	22.9

Table 6. Comparisons of temporal localization mAP on ActivityNet Challenge 2016 [2] test set. The baseline results are quoted from the ActivityNet Challenge 2016 leaderboard [2]. CDC (train) is training the CDC model on the training set only and CDC (train+val) uses the training set and the validation set together to train the CDC model.

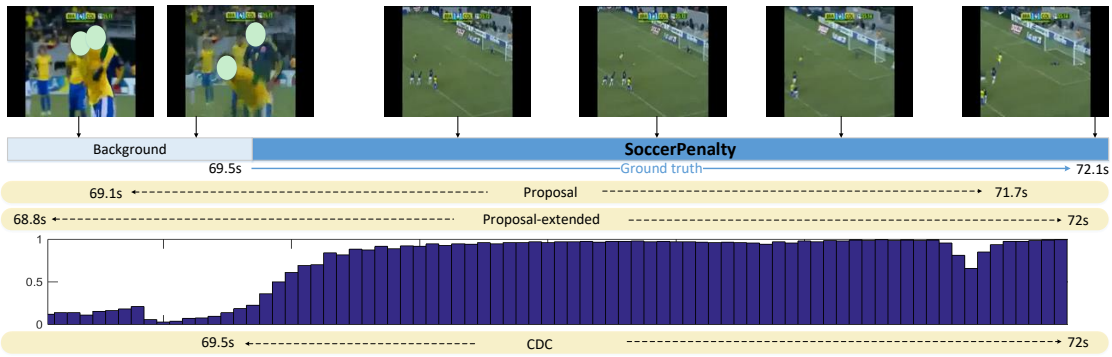

Figure 7. Visualization of the process of refining temporal boundaries for an action proposal segment. Horizontal axis stands for time. From the top to the bottom: (1) frame-level ground truths for a SoccerPenalty action instance in a test video with some representative frames; (2) a corresponding proposal segment; (3) the proposal segment after extension; (4) the per-frame score of being SoccerPenalty predicted by the CDC network; (5) the precisely predicted action instance after the refinement step using CDC.

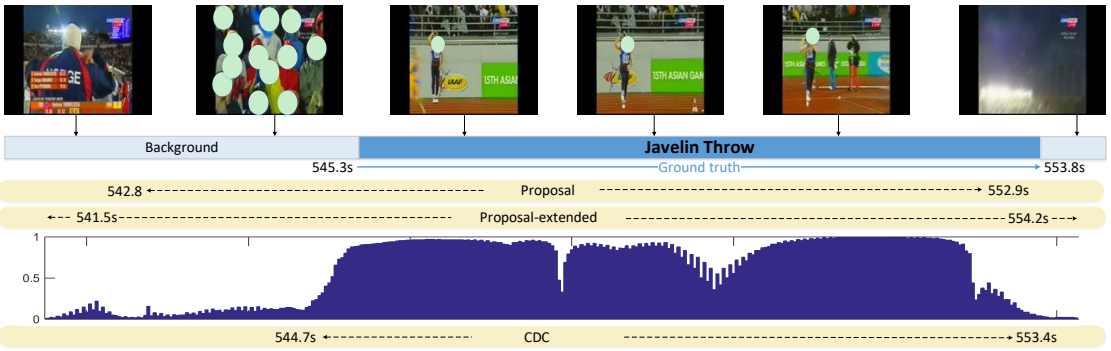

Figure 8. Visualization of the process of refining temporal boundaries for an action proposal segment. Horizontal axis stands for time. From the top to the bottom: (1) frame-level ground truths for a JavelinThrow action instance in a test video with some representative frames; (2) a corresponding proposal segment; (3) the proposal segment after extension; (4) the per-frame score of being JavelinThrow predicted by the CDC network; (5) the precisely predicted action instance after the refinement step using CDC.

Therefore, S-CNN does not suit ActivityNet.

Additional discussions about speed. For the sake of avoiding confusions, we would like to emphasize that the CDC network is end-to-end while the task of temporal localization is not end-to-end due to the need of combing with proposals and performing post-processing. Throughout the paper, the speed is also computed for the CDC network itself. Following C3D [61], each input frame has spatial resolution 128×171 and will be cropped into 112×112 as network input (random cropping during training and center cropping during testing). As indicated in Figure 3, each input video of L frames has the shape of (3, L, 112, 112). As aforementioned, on a single NVIDIA Titan X GPU of 12GB memory, the speed of a CDC network is around 500 Frames Per Second (FPS), which means it can process a 20s long video clip of 25 FPS within one second.

Additional visualization examples. As supplementary material to Figure 4, we provide additional examples to show the process of using Convolutional-De-Convolutional (CDC) model to refine the boundaries of proposal segments and achieve precise temporal action localization on THUMOS'14 [25]. As shown in Figure 7 and Figure 8, the combination of the segment proposal and the CDC frame-level score prediction is powerful. The segment proposal allows for leveraging candidates of coarse boundaries to help handle the noisy outliers in the dipped score intervals such as shown in Figure 8. The proposed CDC model allows for fine-grained predictions at the frame level to help refine the segment boundaries in frame-level for precise localization.

References

[1] Mexaction2. http://mexculture.cnam.fr/xwiki/bin/view/Datasets/Mex+action+dataset, 2015. 2

[2] Activitynet challenge 2016. http://activity-net.org/challenges/2016/, 2016. 8, 10

[3] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. In *ACM Computing Surveys*, 2011. 2

IoU threshold	0.3	0.4	0.5	0.6	0.7
S-CNN [47] w/o CDC	36.3	28.7	19.0	10.3	5.3
ResC3D+S-CNN [47] w/o CDC	40.6	32.6	22.5	12.3	6.4
S-CNN [47]	40.1	29.4	23.3	13.1	7.9
ResC3D+S-CNN [47]	41.3	30.7	24.7	14.3	8.8

Table 7. Temporal action localization mAP on THUMOS'14 as the overlap IoU threshold used in evaluation varies from 0.3 to 0.7. We evaluate our CDC model based on different proposal methods.

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*, 2016. 3

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 3

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. 2016. 3

[7] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 6

[8] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *ICCV*, 2007. 2

[9] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem. Daps: Deep action proposals for action understanding. In *ECCV*, 2016. 1, 3, 6, 7, 10

[10] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 2

[11] A. Gaidon, Z. Harchaoui, and C. Schmid. Actom sequence models for efficient action detection. In *CVPR*, 2011. 2

[12] A. Gaidon, Z. Harchaoui, and C. Schmid. Temporal localization of actions with actoms. In *TPAMI*, 2013. 2

[13] C. Gan, N. Wang, Y. Yang, D.-Y. Yeung, and A. G. Hauptmann. Devnet: A deep event network for multimedia event detection and evidence recounting. In *CVPR*, 2015. 2

[14] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. 2

[15] 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

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[17] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015. 2, 8

[18] F. C. Heilbron, J. C. Niebles, and B. Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *CVPR*, 2016. 1, 2, 7, 10

[19] S. Hong, H. Noh, and B. Han. Decoupled deep neural network for semi-supervised semantic segmentation. In *NIPS*, 2015. 3

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

[21] M. Jain, J. van Gemert, T. Mensink, and C. Snoek. Objects2action: Classifying and localizing actions without any video example. In *ICCV*, 2015. 2

[22] M. Jain, J. van Gemert, and C. Snoek. What do 15,000 object categories tell us about classifying and localizing actions? In *CVPR*, 2015. 2

[23] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010. 2

[24] 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. 5

[25] 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, 2, 6, 11

[26] S. Karaman, L. Seidenari, and A. D. Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. In *ECCV THUMOS Workshop*, 2014. 1, 2, 7

[27] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2, 5

[28] A. Kläser, M. Marszałek, C. Schmid, and A. Zisserman. Human focused action localization in video. In *Trends and Topics in Computer Vision*, 2012. 2

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2

[30] I. Laptev and P. Pérez. Retrieving actions in movies. In *ICCV*, 2007. 2

[31] C. Lea, A. Reiter, R. Vidal, and G. D. Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. In *ECCV*, 2016. 2, 3

[32] G. Lin, C. Shen, A. van den Hengel, and I. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*, 2016. 3

[33] S. Liu, X. Qi, J. Shi, H. Zhang, and J. Jia. Multi-scale patch aggregation (mpa) for simultaneous detection and segmentation. In *CVPR*, 2016. 3

[34] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2, 3, 4

[35] M. A. M. Rohrbach, S. Amin and B. Schiele. A database for fine grained activity detection of cooking activities. In *CVPR*, 2012. 2

[36] P. Mettes, J. van Gemert, and C. Snoek. Spot on: Action localization from pointly-supervised proposals. In *ECCV*, 2016. 1

[37] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. 3

[38] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, 2013. 2

[39] D. Oneata, J. Verbeek, and C. Schmid. The lear submission at thumos 2014. In *ECCV THUMOS Workshop*, 2014. 1, 2, 7

[40] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010. 2

[41] R. Poppe. A survey on vision-based human action recognition. In *Image and vision computing*, 2010. 2

[42] M. M. Puscas, E. Sangineto, D. Culibrk, and N. Sebe. Unsupervised tube extraction using transductive learning and dense trajectories. In *ICCV*, 2015. 2

[43] A. Richard and J. Gall. Temporal action detection using a statistical language model. In *CVPR*, 2016. 1, 2, 7

[44] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 2

[45] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *TPAMI*, 2016. 2, 3, 4

[46] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. *arXiv preprint arXiv:1703.01515*, 2017. 8

[47] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*, 2016. 1, 2, 3, 5, 6, 7, 10, 12

[48] G. A. Sigurdsson, O. Russakovsky, A. Farhadi, I. Laptev, and A. Gupta. Much ado about time: Exhaustive annotation of temporal data. In *HCOMP*, 2016. 2

[49] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016. 2

[50] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2, 6

[51] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 2, 6

[52] B. Singh. Action detection. In *CVPR ActivityNet Workshop*, 2016. 10

[53] 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*, 2016. 2

[54] G. Singh and F. Cuzzolin. Untrimmed classification for activity detection: submission to activitynet challenge. In *CVPR ActivityNet Workshop*, 2016. 1, 2, 10

[55] K. Soomro, H. Idrees, and M. Shah. Action localization in videos through context walk. In *ICCV*, 2015. 2

[56] K. Soomro, H. Idrees, and M. Shah. Predicting the where and what of actors and actions through online action localization. In *CVPR*, 2016. 2

[57] A. Stoian, M. Ferecatu, J. Benois-Pineau, and M. Crucianu. Fast action localization in large scale video archives. In *TCSVT*, 2015. 2

[58] A. Stoian, M. Ferecatu, J. Benois-Pineau, and M. Crucianu. Scalable action localization with kernel-space hashing. In *ICIP*, 2015. 2

[59] W. Sultani and M. Shah. What if we do not have multiple videos of the same action? – video action localization using web images. In *CVPR*, 2016. 2

[60] 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. 3

[61] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 1, 2, 3, 5, 10

[62] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Deep end2end voxel2voxel prediction. In *CVPR Workshop on Deep Learning in Computer Vision*, 2016. 2, 3, 4

[63] J. van Gemert, M. Jain, E. Gati, and C. Snoek. Apt: Action localization proposals from dense trajectories. In *BMVC*, 2015. 2

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

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

[66] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. In *ECCV THUMOS Workshop*, 2014. 1, 2, 7

[67] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 2

[68] R. Wang and D. Tao. Uts at activitynet 2016. In *CVPR ActivityNet Workshop*, 2016. 1, 2, 8, 10

[69] 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. 2

[70] P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. In *ICCV*, 2015. 2

[71] C. Xu and J. J. Corso. Actor-action semantic segmentation with grouping process models. In *CVPR*, 2016. 2

[72] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. In *CVPR*, 2015. 2

[73] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *arXiv preprint arXiv:1507.05738*, 2015. 3, 6, 7

[74] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016. 1, 3, 7

[75] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 3

[76] G. Yu and J. Yuan. Fast action proposals for human action detection and search. In *CVPR*, 2015. 2

[77] J. Yuan, B. Ni, X. Yang, and A. Kassim. Temporal action localization with pyramid of score distribution features. In *CVPR*, 2016. 3, 7

[78] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 3

[79] M. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *CVPR*, 2010. 3

[80] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015. 3