Temporal Deformable Residual Networks for Action Segmentation in Videos

Peng Lei and Sinisa Todorovic
Oregon State University
Corvallis, OR 97331, USA
{leip, sinisa}@oregonstate.edu

Abstract

This paper is about temporal segmentation of human actions in videos. We introduce a new model – temporal deformable residual network (TDRN) – aimed at analyzing video intervals at multiple temporal scales for labeling video frames. Our TDRN computes two parallel temporal streams: i) Residual stream that analyzes video information at its full temporal resolution, and ii) Pooling/unpooling stream that captures long-range video information at different scales. The former facilitates local, fine-scale action segmentation, and the latter uses multiscale context for improving accuracy of frame classification. These two streams are computed by a set of temporal residual modules with deformable convolutions, and fused by temporal residuals at the full video resolution. Our evaluation on the University of Dundee 50 Salads, Georgia Tech Egocentric Activities, and JHU-ISI Gesture and Skill Assessment Working Set demonstrates that TDRN outperforms the state of the art in frame-wise segmentation accuracy, segmental edit score, and segmental overlap F1 score.

1. Introduction

In this paper, we address action segmentation where the goal is to label video frames with appropriate action classes. Action segmentation is a basic vision problem, and of great importance to a wide range of applications, including video surveillance and robot navigation.

Recent approaches typically address this problem in two steps: i) Extraction of spatial or spatiotemporal features using convolutional neural networks, e.g., two-stream CNNs [36] or local 3D ConvNets [43], and ii) Classification of the extracted features using a one-directional model, e.g., encoder-decoder temporal convolutional networks (ED-TCN) [22], or bi-directional LSTM networks (Bi-LSTM) [37, 17]. Although recurrent deep models have shown promise in capturing latent temporal patterns [37, 17, 6], they are hard to train [28], and have a limited span of attention [37].

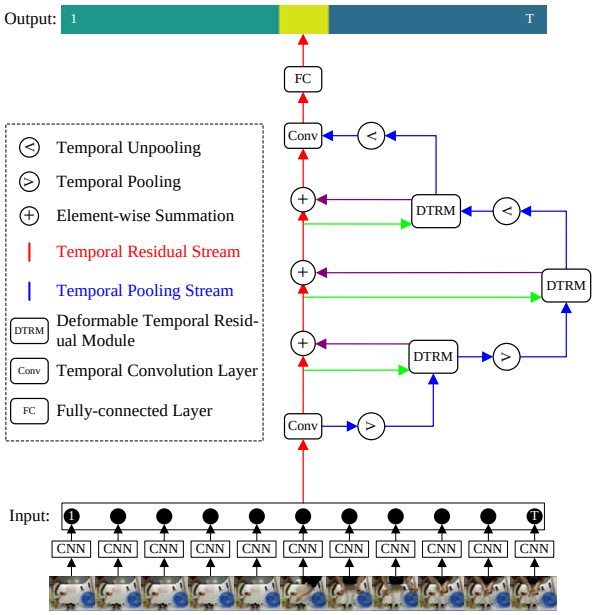


Figure 1: For action segmentation, TDRN takes frame-level CNN features as input and outputs frame-wise action labels. TDRN computes two processing streams: Residual stream (marked red) that analyzes video information at its full temporal resolution for precise action segmentation, and Pooling/unpooling stream (marked blue) that captures temporal context at different scales for accurate action recognition. The two streams are fused through a set of deformable temporal residual modules (DTRMs). Best viewed in color.

Toward overcoming these limitations, we present a new temporal convolutional model, named temporal deformable residual network (TDRN). TDRN classifies every video frame using a deep temporal residual network. The residual network takes frame-level CNN features as input and computes deformable convolutions along time at multiple temporal scales, starting at the frame-level resolution. As shown in Fig. 1, this computation is done in two parallel temporal streams: i) Residual stream that analyzes video information

at its full temporal resolution, and ii) Pooling/unpooling stream that captures long-range video information at different scales. The first stream is aimed at resolving ambiguities about local, frame-to-frame segmentation, and the second stream uses multiscale context for improving accuracy of frame classification. The temporal residual stream and the temporal pooling stream are fused through a set of deformable temporal residual modules (DTRMs), and coupled with temporal residuals at the full temporal resolution. In addition, TDRN computes deformable temporal convolutions for modeling variations in temporal extents of human actions, similar to deformable spatial convolution that has been shown to improve object detection in images [3].

As our results demonstrate, the two-stream residual computation and deformable temporal convolutions make TDRN more robust against temporal transformations than recent deep networks, including encoder-decoder temporal convolutional networks (ED-TCNs) [25, 22], temporal convolutional U-networks (TUNets) [34], and temporal residual networks (TResNets) [15], illustrated in Fig. 2. As can be seen in Fig. 2, ED-TCNs use a sequence of regular temporal convolutions and temporal pooling/unpooling layers within a single processing stream. TUNets simply concatenate features computed in their unpooling path with the corresponding features at the same temporal scale from the pooling path, as indicated by the cyan arrows. TResNets add shortcut connections (marked brown) between a layer and its succeeding layer for allowing the gradients to propagate more effectively through the network in learning. None of these models use deformable temporal convolutions and two processing streams since they all compute regular temporal convolutions in a single processing stream. Unlike these related models, we use two processing streams and deformable temporal convolutions, enabling robust action classification and accurate action segmentation in videos.

We evaluate TDRN on the following benchmark datasets: University of Dundee 50 Salads (50Salads), Georgia Tech Egocentric Activities (GTEA) and JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS). Our results demonstrate that TDRN is capable of accurately capturing action durations and transitions between distinct actions. Also, TDRN outperforms the state of the art in frame-wise segmentation accuracy, segmental edit score, and segmental overlap F1 score.

Our key contributions include:

- A new fully-convolutional temporal residual network that consists of two processing streams aimed at extracting both multiscale temporal abstractions and frame-level features for reliable action recognition and precise action segmentation.
- We are not aware of any prior work that uses deformable temporal convolutions; we show they improve action segmentation over regular temporal convolutions.

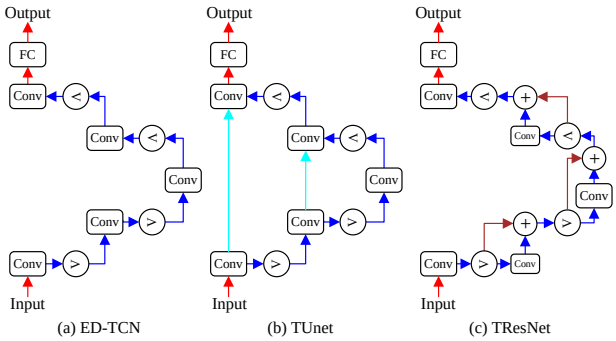


Figure 2: Deep architectures of recent work: (a) Encoder-decoder temporal convolutional networks (ED-TCNs) [25], (b) Temporal convolutional U-networks (TUNets) [34], (c) Temporal residual networks (TResNets) [15]. A comparison of these architectures with our TDRN shown in Fig. 1 makes our differences obvious: none of these models use deformable temporal convolutions and two processing streams. Best viewed in color.

- We outperform the state of the art in action segmentation on the 50Salads, GTEA and JIGSAWS datasets.

2. Related Work

This section reviews the most related work for action segmentation and detection in which most of them are about temporal modeling. A host of work on spatiotemporal modeling for video recognition [41, 19, 45, 43, 36, 51, 2, 10, 11, 13, 46] and action detection [35, 18, 38, 16, 5] are beyond the scope of this paper.

Action Segmentation. Existing approaches typically first extract frame-level features, and then pass them to a temporal model for frame labeling. For example, Yeung et al. [48] use an attention LSTM network to model feature dependencies over a fixed temporal interval. Singh et al. [37] present a multi-stream bi-directional recurrent neural network for fine-grained action detection. Fathi et al. [7, 9, 8] use a segmental model that captures object states at the action's start and end. Richard et al. [32] resort to a statistical language model for representing temporal and contextual structure in videos of varying lengths. Kuehne et al. [21] use Hidden Markov Models (HMMs) on dense-trajectory features and propose an end-to-end generative approach for action segmentation.

The approach of Lea et al. [22] is the most related to ours, as they use two temporal convolutional networks for action segmentation and detection. However, their model computes regular temporal convolutions in a single processing stream, whereas our TDRN computes deformable temporal convolutions in two temporal streams. Ding et al. [6] replace the convolutional decoder in the approach of Lea et al. [22]

with a bi-directional LSTM (Bi-LSTM) [14]. However, their network is a hybrid of temporal convolutional network and temporal recurrent network, and thus inherits the well-known limitations of recurrent models, including difficult training [28], and limited attention span [37].

Action Detection. A number of approaches to action detection is also related to ours. For example, for action detection, (a) Multi-region two-stream network [29] links frame-level detections of the faster R-CNN [31]; (b) Recurrent models are learned from 3D skeleton data [26] and under weak supervision [33]; (c) Reinforcement learning is used for predicting temporal bounds of actions based on observing only a fraction of the video [49]; (d) Structured temporal pyramid models the temporal structure of actions [52]; (e) Region convolutional 3D network (R-C3D) with 3D ROI pooling encodes video streams [47]; (f) Flow network searches for temporal intervals with a maximum sum of frame-wise classification scores [50]; (g) Temporal convolutional model extracts context of action proposals through a pair-wise sampling layer [4]; and (h) Temporal single shot action detector network detects action instances [27].

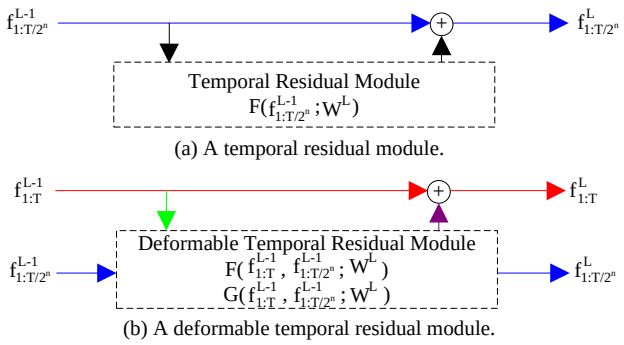
In some of the aforementioned approaches, video features are usually sampled at two temporal scales for generating good action proposals. Also, some of the approaches consist of modules that are typically not jointly trained end-to-end. In contrast, our TDRN fuses multiscale temporal abstractions with features extracted at the frame-wise temporal scale, and can be trained in an end-to-end fashion.

There are some similarities between TDRN and recent work on semantic image segmentation [30], which uses a two-stream spatial residual network to compute pixel-level semantic labeling in the image. In contrast, TDRN computes temporal residual convolutions, which are additionally deformable [3], i.e., capable of modeling variations of temporal extents of actions via deformable temporal convolutions. Hence, we extend [30, 3] from the spatial to temporal domain, where TDRN also analyzes multiple temporal scales.

3. Temporal Deformable Residual Network

TDRN computes the residual and pooling/unpooling streams in parallel. As shown in Fig. 1, features along the residual stream are computed using a sequence of residuals at the full temporal resolution, whereas features in the temporal pooling stream are computed at coarser temporal scales by a sequence of deformable temporal convolutions followed by temporal pooling and corresponding unpooling. The residual stream operates at the finest temporal scale for accurately localizing action boundaries. The temporal pooling/unpooling stream computes contextual features at multiple temporal scales for accurate action recognition using a sequence of deformable temporal residual modules (DTRMs).

Fig. 3 shows key differences between a common temporal residual module and our DTRM. The former has only one input and one output, while DTRM has two inputs and two outputs. This is because DTRM simultaneously operates on both the residual and pooling streams.



(a) A temporal residual module.

(b) A deformable temporal residual module.

Figure 3: Key differences between: (a) Common temporal residual module with a single input and output; and (b) Our deformable temporal residual module (DTRM) with two inputs and two outputs. The input represents feature sequences $f^{L-1}_{1:T/2^n}$ and $f^{L-1}_{1:T}$ with temporal lengths of $T/2^n$ and T, which are computed by the previous layer $L-1$. In (a), the output features are computed by a standard temporal convolution, $F(f^{L-1}_{1:T/2^n}; W^L)$. In (b), the output is computed using a cascade of a deformable temporal convolution, $G(f^{L-1}_{1:T}, f^{L-1}_{1:T/2^n}; W^L)$, followed by a convolution and unpooling, $F(f^{L-1}_{1:T}, f^{L-1}_{1:T/2^n}; W^L)$.

DTRM takes as input two feature sequences, $f^{L-1}_{1:T/2^n}$ and $f^{L-1}_{1:T}$, with temporal lengths of $T/2^n$ and T, which are computed by the previous layer $L-1$ in TDRN. Specifically, $f^{L-1}_{1:T/2^n}$ is produced by the pooling stream and $f^{L-1}_{1:T}$ comes from the residual stream of $L-1$ layer. Note that the layer number L is correlated with the temporal scale n at which the features are computed in TDRN. These are depicted in Fig. 1 as "vertical" and "horizontal" processing levels in TDRN, respectively. For computing the output feature sequences, $f^{L}_{1:T}$ and $f^{L}_{1:T/2^n}$, DTRM uses a deformable temporal convolution, $G(f^{L-1}_{1:T}, f^{L-1}_{1:T/2^n}; W^L)$, followed by a convolution and unpooling, $F(f^{L-1}_{1:T}, f^{L-1}_{1:T/2^n}; W^L)$, as

$$\begin{aligned} f^{L}_{1:T} &= f^{L-1}_{1:T} + F(f^{L-1}_{1:T}, f^{L-1}_{1:T/2^n}; W^L) \\ f^{L}_{1:T/2^n} &= G(f^{L-1}_{1:T}, f^{L-1}_{1:T/2^n}; W^L) \end{aligned} \tag{1}$$

where W^L denotes network parameters. In (1), the output of G is input to F to produce the residual stream.

It is worth noting that our TDRN has similar training characteristics as ResNet [15], since losses can be easily propagated back to the input through the residual stream. In the following section, we describe DTRM in greater detail.

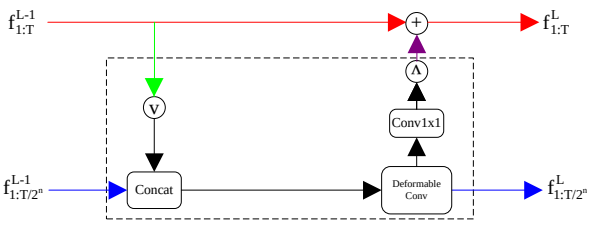


Figure 4: DTRM: Given the residual and pooling features of the previous layer as inputs, DTRM applies pooling to the residual stream and then concatenates the result with the input pooling stream. The concatenated features are then processed by the deformable temporal convolution, resulting in the output pooling features, $f^L_{1:T/2^n}$. Also, a temporal residual is computed from $f^L_{1:T/2^n}$ by the 1×1 temporal convolution and temporal unpooling, resulting in output residual features, $f^L_{1:T}$.

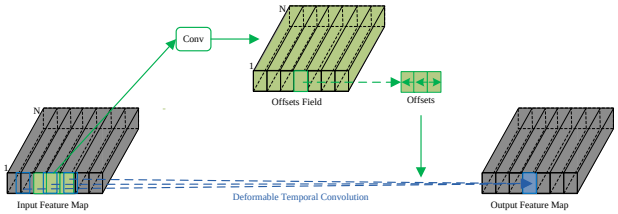


Figure 5: An illustration of deformable temporal convolution with kernel size 3 and dilation size 1. Both the temporal offsets and output features are obtained by applying a temporal convolutional layer over the same input feature maps. The offset fields have the same size as the input feature map.

3.1. Deformable Temporal Residual Module

The pooling and residual streams in our TDRN are fused through a sequence of DTRMs. Fig. 4 illustrates the architecture of DTRM. DTRM takes as input the residual and pooling features of the previous layer, $f^{L-1}_{1:T}$ and $f^{L-1}_{1:T/2^n}$, where the pooling sequence of features is computed at a coarser temporal resolution, n, and hence is 2^n times shorter in time than the residual sequence of features. DTRM first applies temporal pooling to $f^{L-1}_{1:T}$, and then concatenates the result with $f^{L-1}_{1:T/2^n}$. The concatenated features are then processed by the deformable temporal convolution module, explained in greater detail in Sec. 3.2, resulting in the output pooling features, $f^L_{1:T/2^n}$, of the same length as the corresponding input pooling features $f^{L-1}_{1:T/2^n}$. From $f^L_{1:T/2^n}$, DTRM computes output residual features, $f^L_{1:T}$, using the 1×1 temporal convolution and temporal unpooling.

3.2. Deformable Temporal Convolution Module

The deformable temporal convolution module is aimed at improving standard fixed-structure temporal convolutions in modeling temporal variations of action boundaries along the video. It consists of a deformable temporal convolution layer followed by a Normalized Rectified Linear Unit (NRLU) [22], defined as follows:

$$NRLU(\cdot) = \frac{ReLU(\cdot)}{max(ReLU(\cdot)) + \epsilon} \tag{2}$$

where $ReLU$ represents a ReLU layer, $max(\cdot)$ returns the maximal ReLU activation within the layer, and $\epsilon = 0.00001$. Below, we specify deformable temporal convolution.

A temporal convolution can be decomposed into two steps: sampling of input features at specified moments in time, and weighted summation of the sampled features. Analogous to deformable spatial convolutions of objects in images [3], in our approach, the temporal sampling locations that specified by the convolutional kernel are augmented with variable temporal offsets, which in turn are learned end-to-end along with the other network parameters.

Let $\mathcal{I}$ denote the time interval of input feature map f_{in}, and W denote convolution weights. Note that $\mathcal{I}$ defines the temporal receptive field size as well as the dilation size. The temporal convolution consists of sampling over $\mathcal{I}$ and summing the weighted sampled values with weights W as

$$f_{\text{out}}(t_0) = \sum_{t_i \in \mathcal{I}} W(t_i) \cdot f_{in}(t_0 + t_i). \tag{3}$$

Deformable convolution specifies a set of offsets $\triangle = \{\triangle t_i | i = 1, 2, \cdots, |\mathcal{I}|\}$, and augments the temporal sampling in (3) as

$$f^{\text{deform}}_{\text{out}}(t_0) = \sum_{t_i \in \mathcal{I}} W(t_i) \cdot f_{in}(t_0 + t_i + \triangle t_i). \tag{4}$$

From (4), the sampling is defined over variable time moments $t_i + \triangle t_i$.

Note that $\triangle t_i$ is typically learned as a real number. Thus, $t_0 + t_i + \triangle t_i$ is also a real number. We identify the set of nearest integer temporal locations to $t_0 + t_i + \triangle t_i$ in the feature map, and use bilinear temporal interpolation to compute the ith input feature for the summation in (4).

As illustrated in Fig. 5, the temporal offsets $\{\triangle t_i\}$ are obtained by applying a temporal convolutional to the input feature maps. The kernel size and the dilation size of the temporal convolution kernel for computing the offsets are the same as those of the temporal kernel used for computing output features (e.g., 3×1 with dilation 1 in Fig. 5). The resulting offset fields have the same size as the input feature map. During training, both the temporal convolutional kernel for generating the output features and the kernel for generating the offsets are learned end-to-end simultaneously.

Dataset	50Salads (mid)			GTEA			JIGSAWS		
Model	F1@{10,25,50}	Edit	Acc	F1@{10,25,50}	Edit	Acc	F1@{10}	Edit	Acc
ED-TCN [22]	68.0,63.9,52.6	59.8	64.7	72.2,69.3,56.0	-	64.0	89.2	84.7	80.8
TUnet [34]	59.3,55.6,44.8	50.6	60.6	67.1,63.7,51.9	60.3	59.9	85.9	79.8	80.2
TResNet [15]	69.2,65.0,54.4	60.5	66.0	74.1,69.9,57.6	64.4	65.8	86.2	85.2	81.1
TDRN	72.9,68.5,57.2	66.0	68.1	79.2,74.4,62.7	74.1	70.1	92.9	90.2	84.6

Table 1: Performance comparison with respect to the most related temporal convolution models including ED-TCN [22], TUNet [34] and TResNet [15].

Layer/Module	Kernel Specification
FC	Dense(C, 'softmax')
Conv	Conv1D(64, 50, 1, 1)
DTRM	Conv1D(64, 50, 1, 1, Offsets(96, 50, 1, 1))
DTRM	Conv1D(96, 50, 1, 1, Offsets(64, 50, 1, 1))
DTRM	Conv1D(64, 50, 1, 1, Offsets(64, 50, 1, 1))
Conv	Conv1D(64, 50, 1, 1)

Table 2: TDRN architecture: The temporal convolution kernel is described in the same format as in Keras [1], i.e., Conv1D(filters, kernel size, strides, dilation rate). The last argument of a DTRM kernel specifies the temporal convolution kernel corresponding to offsets. C denotes the number of action classes including background class. The fully-connected layer, Dense, is applied to every temporal window of the input video.

4. Network Configurations and Training

Our TDRN consists three DTRMs, which are implemented with Keras [1] and TensorFlow. As input, TDRN uses a set of frame-level video features computed by CNNs. We use the same CNN features as in [22]. The output of TDRN is the sequence of action labels assigned to video frames. A detailed description of each module in TDRN is summarized in Tab. 2. For simplicity, Tab. 2 omits the details of NRLUs that follow every deformable temporal convolution layer in DTRM and every temporal convolution layer in TDRN. They are fully specified in Sec. 3.2.

Parameters of TDRNs are learned using the categorical cross entropy loss with Stochastic Gradient Descent and ADAM [20] step updates. The batch size and the number of epoches are 8 and 200, respectively. Dropouts are also applied in all the temporal convolutional layers.

Note that n does not require careful tuning, because it is not a free parameter, but set to a fixed value that depends on a given network architecture. Specifically, n is uniquely determined by the number of pooling and unpooling operations in the network, as the pooling reduces the temporal length of video processing by a half, and the unpooling doubles the temporal length of video processing. For the model depicted in Fig. 1 and summarized in Tab. 2, there are 3 residual modules (DTRMs) and 2 pooling/unpooling operations, so the values of n in each of the 3 DTRMs, bottom to top, must be 1, 2 and 1, respectively. We use the same architecture and hence the same values of n for all datasets.

5. Experimental Results

We conduct experiments on three challenging action segmentation datasets, including the University of Dundee 50 Salads (50Salads) [40], Georgia Tech Egocentric Activities (GTEA) [9] and the JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) [12]. For evaluation, we use three standard metrics, including F1@k, edit score and accuracy of frame labeling.

5.1. Datasets, Metrics and Baselines

Datasets. The 50Salads contains 50 videos with 25 people preparing salad. Each video contains 9000 to 18000 frames with accelerometer data, depth information, RGB data and action label. As input to TDRN, we use the same spatial CNN features as in [22], where the CNN is trained on RGB images showing 17 mid-level action classes. For evaluation on this dataset, we perform the same 5-fold cross-validation as the state of the art, and report the average results.

The GTEA dataset contains 28 videos of seven fine-grained types of daily activities in a kitchen. An activity is performed by four different subjects and each video contains about 1800 RGB frames, showing a sequence of 20 actions including the background action. For fair comparison, input to TDRN are the same CNN features as those used in [22]. For this dataset, we perform the same 4-fold cross-validation as prior work, and report the average results.

The JIGSAWS dataset contains 39 surgical videos with 10 different actions. Each video contains about 3000 frames showing about 20 actions. For fair comparison, we use the same input features of 39 suturing videos as in [25, 22], and perform the same 8-fold cross-validation as prior work, and report the average results.

As in related work [22], we first downsample the video frames and then handle different lengths of downsampled video clips as follows. We first identify the maximum tem-

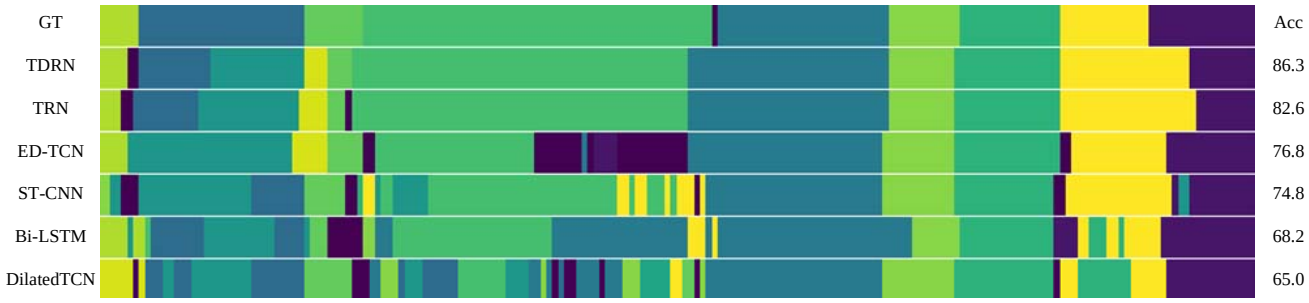


Figure 6: Action segmentations for a sample test video named *rgb-22-2.avi* from the 50 Salads dataset. Top-down, the rows correspond to ground truth sequence of actions {place lettuce into bowl, cut cheese, place cheese into bowl, peel cucumber, background, cut cucumber, place cucumber into bowl, mix ingredients, serve salad onto plate, add dressing}, and predictions of TDRN, TRN, ED-TCN [22], ST-CNN [23], Bi-LSTM [37] and Dilated TCN [22].

poral length in the dataset, and then pad zeros to those clips that are shorter than the maximum length.

Metrics. For all the three datasets, we use the following evaluation metrics as in [22]: frame-wise accuracy, segmental edit score, and segmental overlap F1 score with threshold $k/100$, denoted as F1@k. Frame-wise accuracy is one of the most common evaluation metrics for action segmentation. Its drawback is that it does not take into account the temporal structure of the prediction. Consequently, results with large qualitative differences may have the same frame-wise accuracy. Also, this metric does not capture the case of oversegmentation, when the results do not respect the true temporal continuity of human actions, and yet score high frame-wise accuracy. To address these limitations, evaluations presented in [23, 24] additionally use a segmental edit score, which penalizes oversegmentation. The approach of [22] uses F1@k as a suitable metric for testing both action segmentation and action detection, since it also penalizes oversegmentation errors, but ignores minor temporal shifts between the predictions and ground truth (which might arise from annotation noise). F1@k score is determined by the total number actions but not depend on the duration of each action instance. It is a metric that similar to mean average precision (mAP) with an Intersection-Over-Union (IoU) overlap criterion which is commonly used in object detection.

Baselines. For ablation studies, we specify the following TDRN variants: (1) TRN : TDRN without deformable convolution (i.e., uses a standard temporal convolution); and (2) TDRN+UNet : TDRN with added TUnet connections, marked cyan in Fig. 2. We also compare with the following closely related work: (3) Spatial CNN [23]: Frame-wise classification using CNN features of a single RGB frame that capture object texture and spatial location; (4) ST-CNN [23]: Temporal convolutional filter that builds on top of spatial CNN to capture scene changes over the course of action; (5) Bi-LSTM [37]: Bi-directional temporal LSTM; 6) ED-TCN [22]: encoder-decoder temporal convolution neural network; and 7) Dilated TCN [22]: encoder-decoder temporal convolution neural network with dilated temporal convolution. In addition, we compare with the baselines IDT+LM [32], EgoNet+TDD [39], and MSM-CRF [42] and TCN [25] on 50Salads, GTEA and JIGSAWS, respectively.

50 Salads (mid)	F1@{10,25,50}	Edit	Acc
Spatial CNN [23]	32.3,27.1,18.9	24.8	54.9
IDT+LM [32]	44.4,38.9,27.8	45.8	48.7
Dilated TCN [22]	52.2,47.6,37.4	43.1	59.3
ST-CNN [23]	55.9,49.6,37.1	45.9	59.4
Bi-LSTM [37]	62.6,58.3,47.0	55.6	55.7
ED-TCN [22]	68.0,63.9,52.6	59.8	64.7
TRN	70.2,65.4,56.3	63.7	66.9
TDRN+UNet	69.6,65.0,53.6	62.2	66.1
TDRN	**72.9**,**68.5**,**57.2**	**66.0**	**68.1**

Table 3: Results on 50 Salads (mid).

In our experiments, for fair comparison, we use the same number of temporal convolution layers and kernels as in [22].

5.2. Comparison with Convolution Models

Tab. 1 presents a comparison of TDRN with the most related temporal convolution models, including ED-TCN [22], TUnet [34] and TResNet [15], illustrated in Fig. 2. The table shows that the performance of TUnet is worse than that of ED-TCN, and that TResNet is able to improve over ED-TCN. Our TDRN outperforms the three related models on all of the datasets. This suggests advantages of explicitly computing the pooling and residual feature streams in TDRN for capturing contextual and fine-scale details, respectively, whereas the three related models do not explicitly compute these streams, as depicted in Fig. 2.

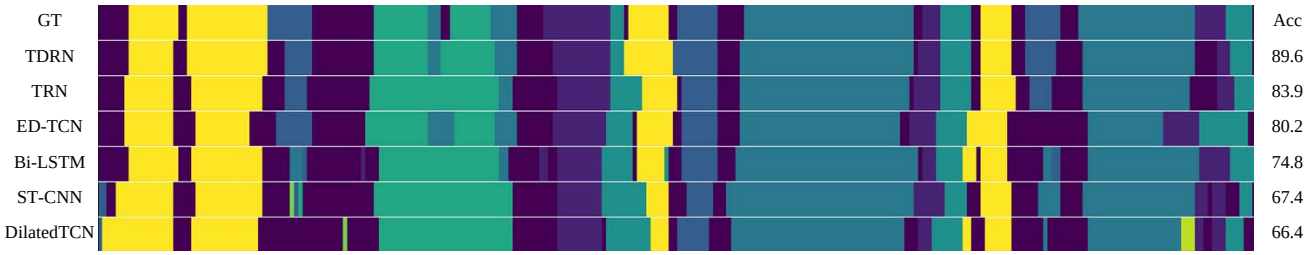


Figure 7: Action segmentations for a sample test video named *S3-CofHoney-C1.mp4* from the GTEA dataset. Top-down, the rows correspond to ground truth sequence of actions { background, take, background, take, open, background, scoop, pour, background, scoop, pour, background, put, close, background, take, background, open, background, pour, background, put, close, background, take, background, open, background, pour, put, close, background, stir }, and predictions of TDRN, TRN, ED-TCN [22], Bi-LSTM [37], ST-CNN [23] and Dilated TCN [22].

GTEA	F1@{10,25,50}	Edit	Acc
Spatial CNN [23]	41.8,36.0,25.1	-	54.1
ST-CNN [23]	58.7,54.4,41.9	-	60.6
Bi-LSTM [37]	66.5,59.0,43.6	-	55.5
Dilated TCN [22]	58.8,52.2,42.2	-	58.3
ED-TCN [22]	72.2,69.3,56.0	-	64.0
EgoNet+TDD [39]	-	-	64.4
TRN	77.4,71.3,59.1	72.2	67.8
TDRN+UNet	78.1,73.8,62.2	73.7	69.3
TDRN	**79.2,74.4,62.7**	**74.1**	**70.1**

Table 4: Results on GTEA.

5.3. Comparison with the State of the Art

50Salads. Tab. 3 presents the results of the state of the art, TDRN and its variants. As can be seen, TDRN gives the best performance, since TDRN accounts for multiscale long-range/high-order temporal dependencies as well as frame-level features. Also, TDRN outperforms its variant TRN which uses a standard temporal convolution, suggesting that TDRN is more robust to temporal variations of action boundaries than TRN. We also observe that augmenting TDRN with UNet-like connections in the variant called TDRN+UNet deteriorates performance of TDRN. This is consistent with the results presented in Sec. 5.2. Fig. 6 qualitatively compares our segmentation results with those of the state of the art on a sample test video from the 50 Salads dataset. As can be seen, TDRN does not suffer from oversegmentation. TDRN produces more accurate action recognition than ED-TCN. For example, in Fig. 6, ED-TCN completely misclassifies the second action in the video, while TDRN is able to generate partially correct prediction. Also, Fig. 6 shows that TDRN predicts more precise action boundaries. This suggests that using deformable temporal convolution is critical for improving accuracy of prediction of action boundaries.

GTEA. Tab. 4 shows that TDRN and its variants achieve superior segmental overlap F1 score, segmental edit score and frame-wise accuracy than the baselines on the GTEA dataset. Among the state of the art, the best accuracy was achieved by the approach of [39], which combines CNN features with trajectory-pooled deep-convolutional descriptors (TDD) [44]. This suggests that our results could be further improved by incorporating the TDD features. Fig. 7 qualitatively compares our segmentation results with those of the state of the art on a sample test video from the GTEA dataset.

JIGSAWS. Tab. 5 compares TDRN with the state of the art on the JIGSAWS dataset. Similar to the results on 50Salads and GTEA, TDRN achieves superior performance in all the three metrics. A qualitative comparison on a sample test video from JIGSAWS is depicted in Fig. 8.

In general, we find that explicit capturing of long-range temporal dependencies by TDRN makes its predictions more reliable than that of the state of the art, especially in cases when distinct actions are visual very similar. For example, in Fig. 6, ED-TCN wrongly predicts the ground truth class *peel cucumber* as *background* while neglecting the temporal dependencies between consecutive action pair {*peel cucumber, cut cucumber*}. On the other hand, TDRN manages to correctly predict *peel cucumber* most likely because it explicitly accounts for long-range action dependencies. We also find similar examples in the results on the GTEA and JIGSAWS datasets (see Fig. 7 and Fig. 8). Empirically, we find that our TDRN sometimes misses the prediction of extremely short action instance that fall in between two long actions, as shown in Fig. 6 and Fig. 7.

5.4. Effect of Kernel Size and Network Depth

We study performance of TDRN as a function of varying kernel size and network depth, i.e., varying temporal receptive size and number of DTRMs. Fig. 9 shows F1@10 score of our TDRN on 50Salads (mid). For all network depths, we observe that the score first increases and then drops as the kernel size becomes larger (i.e., when using longer temporal

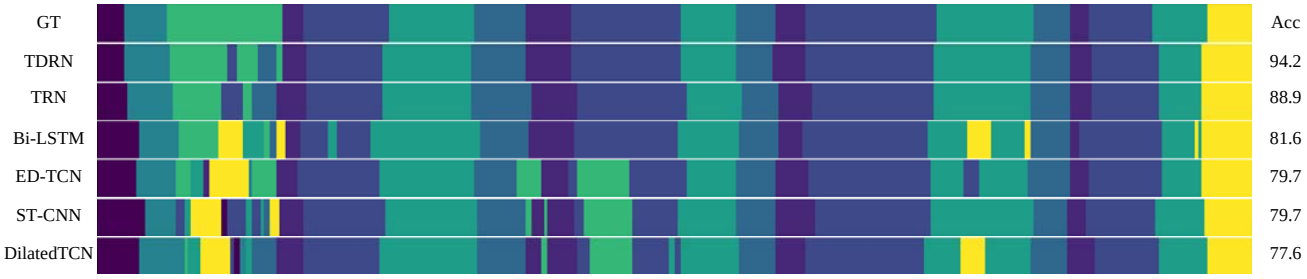


Figure 8: Action segmentations for a sample test video named *Suturing-B002.avi* from the JIGSAWS dataset. Top-down, the rows correspond to ground truth sequence of actions in different gestures {G1, G5, G8, G2, G3, G6, G4, G2, G3, G6, G4, G2, G3, G6, G4, G2, G3, G6, G11}, and predictions of TDRN, TRN, Bi-LSTM [37], ED-TCN [22], ST-CNN [23] and Dilated TCN [22].

JIGSAWS	F1@{10}	Edit	Acc
MSM-CRF [42]	-	-	71.7
Spatial CNN [23]	-	37.7	74.0
ST-CNN [23]	78.3	68.6	78.4
Bi-LSTM [37]	77.8	66.8	77.4
ED-TCN [22]	89.2	84.7	80.8
TCN [25]	-	83.1	81.4
TRN	91.4	87.7	83.3
TDRN+UNet	92.1	89.4	83.9
TDRN	**92.9**	**90.2**	**84.6**

Table 5: Results on JIGSAWS.

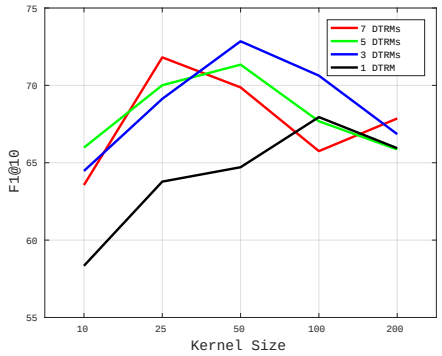


Figure 9: F1@10 of TDRN as a fucntion of temporal kernel size and network depth on 50Salads (mid).

context). This suggests the importance of selecting a suitable kernel size. Our TDRN achieves the best score when the number of DTRMs is 3 (i.e., network depth 6 as specified in Tab. 2) and the kernel size is 50 on the 50Salad dataset. Our optimal network depth agrees with that in [22]. That is, we use the same architecture as specified in Tab. 2 for all datasets for fair comparison with [22].

Note that TDRN training takes 10x less time than for Bi-LSTM, on a Telsa K80 Nvidia gpu card. This is due to independent activations within each temporal convolution layer in TDRN while the activations within Bi-LSTM depend on its previous activations. Hence, for our TDRN, operations can be computed simultaneously in batches.

6. Conclusion

We have presented a new deep architecture, called temporal deformable convolution neural network (TDRN), for action segmentation in videos. TDRN consists of two processing streams: a temporal pooling stream that captures long-range and high-level features at multiple temporal scales, and a temporal residual stream that computes features at the same frame-level temporal resolution as the input video. As the pooling stream accounts for temporal context, it is aimed at improving action recognition. The residual stream is aimed at improving localization of action boundaries. The two streams are aggregated by a cascade of deformable temporal residual modules, each computing deformable temporal convolutions for modeling temporal variations in action boundaries.

Our empirical evaluation on the benchmark University of Dundee 50 Salads (50Salads), Georgia Tech Egocentric Activities (GTEA) and JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) demonstrates that TDRN outperforms the state-of-the-art convolution and temporal convolution models. TDRN produces more accurate action boundary detections, which suggest advantages of our end-to-end learning of deformable temporal convolution over using the standard temporal convolution. Also, TDRN's results tend to better respect common-sense temporal arrangement of actions, due to its explicit learning of long-range temporal dependencies. We have empirically found that TDRN sometimes poorly segments very short actions that fall in between two long actions.

Acknowledgement. This work was supported in part by DARPA XAI Award N66001-17-2-4029.

References

[1] F. Chollet. Keras (2015). *http://keras.io*, 2017. 5

[2] I. Cosmin Duta, B. Ionescu, K. Aizawa, and N. Sebe. Spatio-temporal vector of locally max pooled features for action recognition in videos. In *CVPR*, pages 3097–3106, 2017. 2

[3] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017. 2, 3, 4

[4] X. Dai, B. Singh, G. Zhang, L. S. Davis, and Y. Q. Chen. Temporal context network for activity localization in videos. In *ICCV*, 2017. 3

[5] A. Dave, O. Russakovsky, and D. Ramanan. Predictive-corrective networks for action detection. In *CVPR*, 2017. 2

[6] L. Ding and C. Xu. Tricornet: A hybrid temporal convolutional and recurrent network for video action segmentation. *arXiv preprint arXiv:1705.07818*, 2017. 1, 2

[7] A. Fathi, A. Farhadi, and J. M. Rehg. Understanding egocentric activities. In *ICCV*, pages 407–414. IEEE, 2011. 2

[8] A. Fathi and J. M. Rehg. Modeling actions through state changes. In *CVPR*, pages 2579–2586, 2013. 2

[9] A. Fathi, X. Ren, and J. M. Rehg. Learning to recognize objects in egocentric activities. In *CVPR*, pages 3281–3288. IEEE, 2011. 2, 5

[10] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, pages 3468–3476, 2016. 2

[11] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *CVPR*, pages 4768–4777, 2017. 2

[12] Y. Gao, S. S. Vedula, C. E. Reiley, N. Ahmidi, B. Varadarajan, H. C. Lin, L. Tao, L. Zappella, B. Béjar, D. D. Yuh, et al. Jhu-isi gesture and skill assessment working set (jigsaws): A surgical activity dataset for human motion modeling. In *MICCAI Workshop*, volume 3, 2014. 5

[13] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017. 2

[14] A. Graves, S. Fernández, and J. Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005*, pages 753–753, 2005. 3

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 2, 3, 5, 6

[16] R. Hou, C. Chen, and M. Shah. Tube convolutional neural network (t-cnn) for action detection in videos. In *ICCV*, 2017. 2

[17] D.-A. Huang, L. Fei-Fei, and J. C. Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *ECCV*, pages 137–153. Springer, 2016. 1

[18] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action tubelet detector for spatio-temporal action localization. In *ICCV*, 2017. 2

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

[20] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[21] H. Kuehne, J. Gall, and T. Serre. An end-to-end generative framework for video segmentation and recognition. In *WACV*, pages 1–8. IEEE, 2016. 2

[22] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks for action segmentation and detection. In *CVPR*, 2017. 1, 2, 4, 5, 6, 7, 8

[23] C. Lea, A. Reiter, R. Vidal, and G. D. Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. In *ECCV*, pages 36–52. Springer, 2016. 6, 7, 8

[24] C. Lea, R. Vidal, and G. D. Hager. Learning convolutional action primitives for fine-grained action recognition. In *ICRA*, pages 1642–1649. IEEE, 2016. 6

[25] C. Lea, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks: A unified approach to action segmentation. In *ECCV Workshops*, pages 47–54. Springer, 2016. 2, 5, 6, 8

[26] Y. Li, C. Lan, J. Xing, W. Zeng, C. Yuan, and J. Liu. Online human action detection using joint classification-regression recurrent neural networks. In *ECCV*, pages 203–220. Springer, 2016. 3

[27] T. Lin, X. Zhao, and Z. Shou. Single shot temporal action detection. In *MM*, 2017. 3

[28] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *ICML*, pages 1310–1318, 2013. 1, 3

[29] X. Peng and C. Schmid. Multi-region two-stream r-cnn for action detection. In *ECCV*, pages 744–759. Springer, 2016. 3

[30] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *CVPR*, 2017. 3

[31] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 3

[32] A. Richard and J. Gall. Temporal action detection using a statistical language model. In *CVPR*, pages 3131–3140, 2016. 2, 6

[33] A. Richard, H. Kuehne, and J. Gall. Weakly supervised action learning with rnn based fine-to-coarse modeling. In *CVPR*, 2017. 3

[34] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015. 2, 5, 6

[35] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *CVPR*, 2017. 2

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

[37] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *CVPR*, pages 1961–1970, 2016. 1, 2, 3, 6, 7, 8

[38] G. Singh, S. Saha, and F. Cuzzolin. Online real time multiple spatiotemporal action localisation and prediction on a single platform. In *ICCV*, 2017. 2

[39] S. Singh, C. Arora, and C. Jawahar. First person action recognition using deep learned descriptors. In *CVPR*, pages 2620–2628, 2016. 6, 7

[40] S. Stein and S. J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Ubicomp*, pages 729–738. ACM, 2013. 5

[41] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, pages 4597–4605, 2015. 2

[42] L. Tao, L. Zappella, G. D. Hager, and R. Vidal. Surgical gesture segmentation and recognition. In *MICCAI*, pages 339–346. Springer, 2013. 6, 8

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

[44] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, pages 4305–4314, 2015. 7

[45] L. Wang, Y. Xiong, D. Lin, and L. Van Gool. Untrimmednets for weakly supervised action recognition and detection. In *CVPR*, pages 4325–4334, 2017. 2

[46] Y. Wang, M. Long, J. Wang, and P. S. Yu. Spatiotemporal pyramid network for video action recognition. In *CVPR*, pages 1529–1538, 2017. 2

[47] H. Xu, A. Das, and K. Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *ICCV*, 2017. 3

[48] 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. *International Journal of Computer Vision*, pages 1–15, 2015. 2

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

[50] Z. Yuan, J. C. Stroud, T. Lu, and J. Deng. Temporal action localization by structured maximal sums. In *CVPR*, 2017. 3

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

[52] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, D. Lin, and X. Tang. Temporal action detection with structured segment networks. In *ICCV*, 2017. 3