Temporal Convolutional Networks: A Unified Approach to Action Segmentation

Colin Lea René Vidal Austin Reiter Gregory D. Hager
Johns Hopkins University
{clea1@, rvidal@cis., areiter@cs., hager@cs.}jhu.edu

Abstract

The dominant paradigm for video-based action segmentation is composed of two steps: first, for each frame, compute low-level features using Dense Trajectories or a Convolutional Neural Network that encode spatiotemporal information locally, and second, input these features into a classifier that captures high-level temporal relationships, such as a Recurrent Neural Network (RNN). While often effective, this decoupling requires specifying two separate models, each with their own complexities, and prevents capturing more nuanced long-range spatiotemporal relationships. We propose a unified approach, as demonstrated by our Temporal Convolutional Network (TCN), that hierarchically captures relationships at low-, intermediate-, and high-level time-scales. Our model achieves superior or competitive performance using video or sensor data on three public action segmentation datasets and can be trained in a fraction of the time it takes to train an RNN.

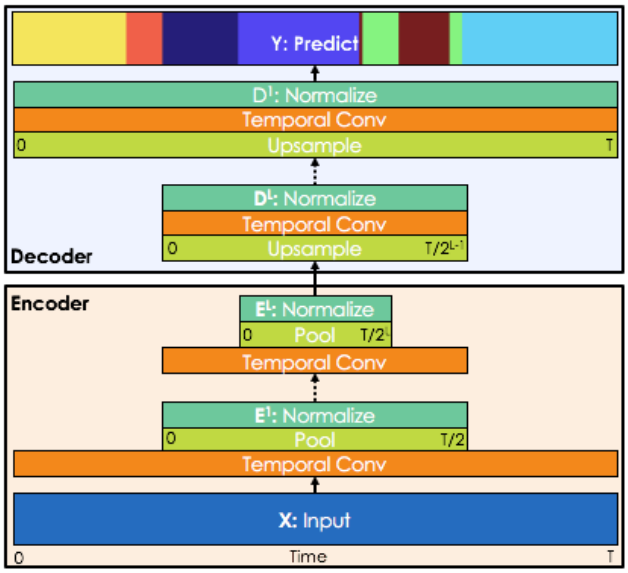

Figure 1. Our temporal encoder-decoder network hierarchically models actions from video or other time-series data.

1. Introduction

Action segmentation is crucial for numerous applications ranging from collaborative robotics to modeling activities of daily living. Given a video, the goal is to simultaneously segment every action in time and classify each constituent segment. While recent work has shown strong improvements on this task, models tend to decouple low-level feature representations from high-level temporal models. Within video analysis, these low-level features may be computed by pooling handcrafted features (e.g. Improved Dense Trajectories (IDT) [21]) or concatenating learned features (e.g. Spatiotemporal Convolutional Neural Networks (ST-CNN) [8, 12]) over a short period of time. High-level temporal classifiers capture a local history of these low-level features. In a Conditional Random Field (CRF), the action prediction at one time step is are often a function of the prediction at the previous time step, and in a Recurrent Neural Network (RNN), the predictions are a function of a set of latent states at each time step, where the latent states are connected across time. This two-step paradigm has been around for decades (e.g., [6]) and typically goes unquestioned. However, we posit that valuable information is lost between steps.

In this work, we introduce a unified approach to action segmentation that uses a single set of computational mechanisms – 1D convolutions, pooling, and channel-wise normalization – to hierarchically capture low-, intermediate-, and high-level temporal information. For each layer, 1D convolutions capture how features at lower levels change over time, pooling enables efficient computation of long-range temporal patterns, and normalization improves robustness towards various environmental conditions. In contrast with RNN-based models, which compute a set of latent activations that are updated sequentially per-frame, we compute a set of latent activations that are updated hierarchically per-layer. As a byproduct, our model takes much

less time to train. Our model can be viewed as a generalization of the recent ST-CNN [8] and is more similar to recent models for semantic segmentation than it is to models for video-analysis. We show this approach is broadly applicable to video and other types of robot sensors.

Prior Work: Due to space limitations, here we will only briefly describe models for time-series and semantic segmentation. See [8] for related work on action segmentation or [20] for a broader overview on action recognition.

RNNs and CRFs are popular high-level temporal classifiers. RNN variations, including Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU), model hidden temporal states via internal gating mechanisms. However, they are hard to introspect and difficult to correctly train [13]. It has been shown that in practice LSTM only keeps a memory of about 4 seconds on some video-based action segmentation datasets [15]. CRFs typically model pairwise transitions between the labels or latent states (e.g., [8]), which are easy to interpret, but over-simplify the temporal dynamics of complex actions. Both of these models suffer from the same fundamental issue: intermediate activations are typically a function of the low-level features at the current time step and the state at the previous time step. Our temporal convolutional filters are a function of raw data across a much longer period of time.

Until recently, the dominant paradigm for semantic semantic was similar to that of action segmentation. Approaches typically combined low-level texture features (e.g., TextonBoost) with high-level spatial models (e.g., grid-based CRFs) that model the relationships between different regions of an image [7]. This is similar to action segmentation where low-level spatiotemporal features are used in tandem with high-level temporal models. Recently, with the introduction of Fully Convolutional Networks (FCNs), the dominant semantic segmentation paradigm has started to change. Long *et al.* [11] introduced the first FCN, which leverages typical classification CNNs like AlexNet, to compute per-pixel object labels. This is done by intelligently upsampling the intermediate activations in each region of an image. Our model is more similar to the recent encoder-decoder network by Badrinarayanan *et al.* [1]. Their encoder step uses the first half of a VGG-like network to capture patterns in different regions of an image and their decoder step takes the activations from the encoder, which are of a reduced image resolution, and uses convolutional filters to upsample back to the original image size. In subsequent sections we describe our temporal variation in detail.

2. Temporal Convolutional Networks (TCN)

The input to our Temporal Convolutional Network can be a sensor signal (e.g. accelerometers) or latent encoding of a spatial CNN applied to each frame. Let $X_t \in \mathbb{R}^{F_0}$ be the input feature vector of length F_0 for time step t for $0 < t \leq T$. Note that the time T may vary for each sequence, and we denote the number of time steps in each layer as T_l. The true action label for each frame is given by $y_t \in \{1, \ldots, C\}$, where C is the number of classes.

Our encoder-decoder framework, as depicted in Figure 1, is composed of temporal convolutions, 1D pooling/upsampling, and channel-wise normalization layers.

For each of the L convolutional layers in the encoder, we apply a set of 1D filters that capture how the input signals evolve over the course of an action. The filters for each layer are parameterized by tensor $W^{(l)} \in \mathbb{R}^{F_l \times d \times F_{l-1}}$ and biases $b^{(l)} \in \mathbb{R}^{F_l}$, where $l \in \{1, \ldots, L\}$ is the layer index and d is the filter duration. For the l-th layer of the encoder, the i-th component of the (unnormalized) activation $\hat{E}_t^{(l)} \in \mathbb{R}^{F_l}$ is a function of the incoming (normalized) activation matrix $E^{(l-1)} \in \mathbb{R}^{F_{l-1} \times T_{l-1}}$ from the previous layer

$$\hat{E}_{i,t}^{(l)} = f(b_i^{(l)} + \sum_{t'=1}^{d} \langle W_{i,t',\cdot}^{(l)}, E_{\cdot,t+d-t'}^{(l-1)} \rangle) \tag{1}$$

for each time t where $f(\cdot)$ is a Leaky Rectified Linear Unit. The normalization process is described below.

Max pooling is applied with width 2 across time (in 1D) such that $T_l = \frac{1}{2} T_{l-1}$.[1] Pooling enables us to efficiently compute activations over a long period of time.

We apply channel-wise normalization after each pooling step in the encoder. This has been effective in recent CNN methods including Trajectory-Pooled Deep-Convolutional Descriptors (TDD) [10]. We normalize the pooled activation vector $\hat{E}_t^{(l)}$ by the highest response at that time step, $m = \max_i \hat{E}_{i,t}^{(l)}$, with some small $\epsilon = 1\text{E-}5$ such that

$$E_t^{(l)} = \frac{1}{m + \epsilon} \hat{E}_t^{(l)}. \tag{2}$$

Our decoder is similar to the encoder, except that upsampling is used instead of pooling, and the order of the operations is now upsample, convolve, then normalize. Upsampling is performed by simply repeating each entry twice.

The probability that frame t corresponds to one of the C action classes is predicted by vector $\hat{Y}_t \in [0, 1]^C$ using weight matrix $U \in \mathbb{R}^{C \times F_0}$ and bias $c \in \mathbb{R}^C$

$$\hat{Y}_t = \text{softmax}(U D_t^{(1)} + c). \tag{3}$$

We explored many other mechanisms, such as adding skip connections between layers, using different patterns of convolutional layers, and other normalization schemes. These helped at times and hurt in others. The aforementioned solution was superior in aggregate.

[1] In theory, this implies T must divisible by 2^L. In practice, we pad each sequence to be of an appropriate length, given the pooling operations, such that the input length of the whole sequence, T, and the length of the output predictions are the same.

Implementation details: Each of the $L = 3$ layers has $F_l = \{32, 64, 96\}$ filters. Filter duration, d, is set as the mean segment duration for the shortest class from the training set. For example, $d = 10$ seconds for 50 Salads. Parameters of our model were learned using the cross entropy loss with Stochastic Gradient Descent and ADAM step updates. All models were implemented using Keras and TensorFlow.

For each frame in our video experiments, the input, X_t, is the first fully connected layer computed in a spatial CNN trained solely on each dataset. We trained the model of [8], except instead of using Motion History Images (MHI) as input to the CNN, we concatenate the following for image I_t at frame t: $[I_t, I_{t-d} - I_t, I_{t+d} - I_t, I_{t-2d} - I_t, I_{t+2d} - I_t]$ for $d = 0.5$ seconds. In our experiments, these difference images – which are a simple type of attention mechanism – tend to perform better than MHI or optical flow across these datasets. Furthermore, for each time step, we perform channel-wise normalization before feeding it into the TCN. This helps with large environmental fluctuations, such as changes in lighting.

3. Evaluation

We evaluate on three public datasets that contain action segmentation labels, video, and in two cases sensor data.

University of Dundee 50 Salads [18] contains 50 sequences of users making a salad. Each video is 5-10 minutes in duration and contains around 30 action instances such as `cutting a tomato` or `peeling a cucumber`. This dataset includes video and synchronized accelerometers attached to ten objects in the scene, such as the *bowl*, *knife*, and *plate*. We performed cross validation with 5 splits on the "eval" action granularity which includes 10 action classes. Our sensor results used the features from [9] which are the absolute values of accelerometer values. Previous results (e.g., [9, 14]) were evaluated using different setups. For example, [9] smoothed out short interstitial background segments. We reran all results to be consistent with [14]. We also included an LSTM baseline for comparison which uses 64 hidden states.

JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) [5] was introduced to improve quantitative evaluation of robotic surgery training tasks. We used Leave One User Out cross validation on the suturing activity, which consists of 39 sequences performed by 8 users about 5 times each. The dataset includes video and synchronized robot kinematics (position, velocity, and gripper angle) for each robot end effector as well as corresponding action labels with 10 action classes. Sequences are a few minutes long and typically contain around 20 action instances.

Georgia Tech Egocentric Activities (GTEA) [4] contains 28 videos of 7 kitchen activities including making a sandwich and making coffee. For each of the four subjects, there is one instance of each activity. The camera is mounted on the head of the user and is pointing at the area in front them. On average there are about 30 actions per video and videos are around a minute long. We used the 11 action classes defined in [3] and evaluated using leave one user out. We show results for user 2 to be consistent with [3] and [16].

Metrics: We evaluated using accuracy, which is simply the percent of correctly labeled frames, and segmental edit distance [9], which measures the correctness of the predicted temporal ordering of actions. This edit score is computed by applying the Levenstein distance to the segmented predictions (e.g. $AAABBA \rightarrow ABA$). This is normalized to be in the range 0 to 100 such that higher is better.

4. Experiments and Discussion

Table 1 includes results for all datasets and corresponding sensing modalities. We include results from the spatial CNN which is input into the TCN, the Spatiotemporal CNN of Lea *et al*. [8] applied to the spatial features, and our TCN.

One of the most interesting findings is that some layers of convolutional filters appear to learn temporal shifts. There are certain actions in each dataset which are not easy to distinguish given the sensor data. By visualizing the activations for each layer, we found our model surmounts this issue by learning temporal offsets from activations in the previous layer. In addition, we find that despite the fact that we do not use a traditional temporal model, such as an RNN or CRF, our predictions do not suffer as heavily from issues like over-segmentation. This is highlighted by the large increase in edit score on most experiments.

Richard *et al*. [14] evaluated their model on the mid-level action granularity of 50 Salads which has 17 action classes. Their model achieved 54.2% accuracy, 44.8% edit, 0.379 mAP IoU overlap with a threshold of 0.1, and 0.229 mAP with a threshold of 0.5.[2] Our model achieves 59.7% accuracy, 47.3% edit, 0.579 mAP at 0.1, and 0.378 mAP at 0.5.

On GTEA, Singh *et al*. [16] reported 64.4% accuracy by performing cross validation on users 1 through 3. We achieve 62.5% using this setup. We found performance of our model has high variance between different trials on GTEA – even with the same hyper parameters – thus, the difference in accuracy is not likely to be statistically significant. Our approach could be used in tandem with features from Singh *et al*. to achieve superior performance.

Our model can be trained much faster than an RNN-LSTM. Using an Nvidia Titan X, it takes on the order of a minute to train a TCN for each split, whereas it takes on the order of an hour to train an RNN-LSTM. The speedup comes from the fact that we compute one set of convolutions for each layer, whereas RNN-LSTM effectively computes one set of convolutions for each time step.

[2] We computed our metrics using the predictions given by the authors.

50 Salads ("eval" setup)

Sensor-based	Edit	Acc
[9] LC-SC-CRF	50.2	77.8
LSTM	54.5	73.3
TCN	**65.6**	**82.0**
Video-based	**Edit**	**Acc**
[8] VGG	7.6	38.3
[8] IDT	16.8	54.3
[8] Seg-ST-CNN	**62.0**	72.0
Spatial CNN	28.4	68.6
ST-CNN	55.5	74.2
TCN	61.1	**74.4**

GTEA

Video-based	Edit	Acc
[3] Hand-crafted	-	47.7
[16] EgoNet	-	57.6
[16] TDD	-	59.5
[16] EgoNet+TDD	-	**68.5**
Spatial CNN	36.6	56.1
ST-CNN	53.4	64.5
TCN	**58.8**	66.1

JIGSAWS

Sensor-based	Edit	Acc
[2] LSTM	75.3	80.5
[9] LC-SC-CRF	76.8	**83.4**
[2] Bidir LSTM	81.1	83.3
[17] SD-SDL	83.3	78.6
TCN	**85.8**	79.6
Vision-based	**Edit**	**Acc**
[19] MsM-CRF	-	71.7
[8] IDT	8.5	53.9
[8] VGG	24.3	45.9
[8] Seg-ST-CNN	66.6	74.7
Spatial CNN	37.7	74.0
ST-CNN	68.0	77.7
TCN	**83.1**	**81.4**

Table 1. Results on 50 Salads, Georgia Tech Egocentric Activities, and JHU-ISI Gesture and Skill Assessment Working Set. Notes: (1) Results using VGG and Improved Dense Trajectories (IDT) were intentionally computed without a temporal component for ablative analysis, hence their low edit scores. (2) We re-computed [9] using the author's public code to be consistent with the setup of [14].

Conclusion: We introduced a model for action segmentation that learns a hierarchy of intermediate feature representations, which contrasts with the traditional low- versus high-level paradigm. This model achieves competitive or superior performance on several datasets and can be trained much more quickly than other models. A future version of this manuscript will include more comparisons and insights on the TCN.

References

[1] V. Badrinarayanan, A. Handa, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*, 2015. 2

[2] R. DiPietro, C. Lea, A. Malpani, N. Ahmidi, S. S. Vedula, G. I. Lee, M. R. Lee, and G. D. Hager. Recognizing surgical activities with recurrent neural networks. In *MICCAI*, 2016. 4

[3] A. Fathi, A. Farhadi, and J. M. Rehg. Understanding egocentric activities. In *ICCV*, 2011. 3, 4

[4] A. Fathi, R. Xiaofeng, and J. M. Rehg. Learning to recognize objects in egocentric activities. In *CVPR*, 2011. 3

[5] 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: M2CAI*, 2014. 3

[6] F. G. Hofmann, P. Heyer, and G. Hommel. Velocity profile based recognition of dynamic gestures with discrete hidden markov models. In *International Workshop on Gesture and Sign Language in Human-Computer Interaction*, 1998. 1

[7] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *NIPS*, 2011. 2

[8] C. Lea, A. Reiter, R. Vidal, and G. D. Hager. Segmental spatio-temporal CNNs for fine-grained action segmentation. *ECCV*, 2016. 1, 2, 3, 4

[9] C. Lea, R. Vidal, and G. D. Hager. Learning convolutional action primitives for fine-grained action recognition. In *ICRA*, 2016. 3, 4

[10] Y. Q. Limin Wang and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 2

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

[12] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 1

[13] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013. 2

[14] A. Richard and J. Gall. Temporal action detection using a statistical language model. In *CVPR*, 2016. 3, 4

[15] 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

[16] S. Singh, C. Arora, and C. V. Jawahar. First person action recognition using deep learned descriptors. In *CVPR*, June 2016. 3, 4

[17] S. Stefati, N. Cowan, and R. Vidal. Learning shared, discriminative dictionaries for surgical gesture segmentation and classification. In *MICCAI Workshop: M2CAI*, 2015. 4

[18] S. Stein and S. J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *UbiComp*, 2013. 3

[19] L. Tao, L. Zappella, G. D. Hager, and R. Vidal. Surgical gesture segmentation and recognition. In *MICCAI*, 2013. 4

[20] M. Vrigkas, C. Nikou, and I. Kakadiaris. A review of human activity recognition methods. *Frontiers in Robotics and AI*, 2015. 2

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