
Temporal attention filters for human activity recognition in videos

AJ Piergiovanni*, Chenyou Fan*, and M. S. Ryoo

School of Informatics and Computing, Indiana University, Bloomington, IN 47408
mryoo@indiana.edu

Abstract

In this paper, we newly introduce the concept of *temporal attention filters*, and describe how they can be used for human activity recognition from videos. Many high-level activities are often composed of multiple temporal parts (e.g., sub-events) with different duration/speed, and our objective is to make the model explicitly consider such temporal structure using multiple temporal filters. Our attention filters are designed to be fully differentiable, allowing end-of-end training of the temporal filters together with the underlying frame-based or segment-based convolutional neural network architectures. The paper not only presents an approach of learning optimal static temporal attention filters to be shared across different videos, but also describes an approach of dynamically adjusting attention filters per testing video using recurrent long short-term memory networks (LSTMs). We experimentally confirm that the proposed concept of temporal attention filters benefits the activity recognition tasks by capturing the temporal structure in videos.

1 Introduction

Human activity recognition is the problem of identifying events performed by humans given a video input. It is formulated as a binary (or multiclass) classification problem of outputting activity class labels, and researchers have been studying better features, representations, and learning algorithms to improve the classification [1]. Such classification not only allows categorization of videos pre-segmented to contain one single activity, but also enables the ‘detection’ of activities from streaming videos [10] together with temporal window proposal methods like the sliding window or selective search. Activity recognition is an important problem with many societal applications including smart surveillance, video search/retrieval, intelligent robots, and other monitoring systems.

Particularly, in the past 2-3 years, activity recognition approaches taking advantage of convolutional neural networks (CNNs) have received a great amount of attention. Motivated by the success of image-based object recognition using CNNs, researchers attempted developing CNNs for videos. Some approaches directly took advantage of image-based CNN architectures by applying them to every video frame [4, 8], while some tried to learn 3-D XYT spatio-temporal convolutional filters from short video segments [15]. In order to represent each video, temporal pooling (e.g., max/average pooling) were often applied on top of multiple (sampled) per-frame or per-video-segment CNNs [4, 5, 14, 8, 15]. Similar to the object recognition, these approaches obtained superior results compared to traditional approaches of using hand-crafted features.

However, in terms of considering activities’ temporal structure in videos, previous CNN approaches were limited. Many high-level activities are often composed of multiple temporal parts (e.g., sub-events) with different duration/speed, but the previous approaches mostly ignored such aspect. For instance, the typical strategy of taking max (or average) pooling over sampled per-frame or per-segment CNN responses (e.g., [4, 14, 8, 15]) completely ignores such temporal structure in longer

*These authors contributed equally to the paper.

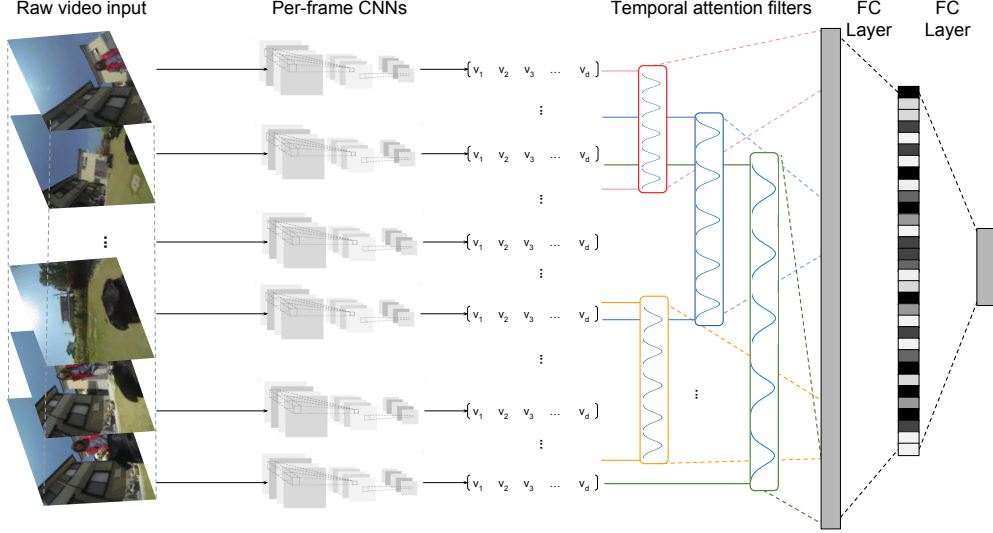


Figure 1: Illustration of our overall recognition architecture with temporal attention filters. M number of temporal filters are *learned* to focus on different temporal part of video frame features. Each filter is composed of a set of Gaussian filters which take a weighted sum of local information. Outputs of the temporal filters are concatenated, and attached with a fully connected layers to perform video classification.

activity videos. What we need instead is an approach that explicitly ‘learns’ to focus on important sub-intervals of the activity videos while also optimizing their temporal resolution for the recognition. [12] showed a potential that making the system consider multiple video sub-intervals using a temporal pyramid benefits the recognition, but it was done with predetermined intervals without any learning.

This paper presents a new video classification approach that overcomes such limitation using temporal attention filters. We newly introduce the concept of fully differentiable *temporal attention filters* and describe how they can be learned and used to enable better recognition of human activities from videos. The main idea is to make the network learn and take advantage of multiple temporal attention filters to be applied on top of per-frame CNNs (Figure 1). Each learned attention filter corresponds to a particular sub-interval of the activity the system should focus on, and is represented with its center location, duration, and resolution in the relative temporal coordinate. We define the ‘read’ operation of abstracting per-frame (or per-segment) CNN responses within the sub-interval corresponding to the attention filter, allowing the system to use their results for the classification. Notably, our temporal attention filters are designed to be fully differentiable, motivated by the spatial attention filters for images [2]. This allows the *end-to-end* training of the parameters deciding attention filters; the system learns temporal attention filters jointly with the parameters of the underlying per-frame (or per-segment) CNNs.

The paper not only presents an approach of learning optimal static temporal attention filters to be shared across different videos, but also present an approach of dynamically adjusting attention filters per testing video using recurrent long short-term memory cells (LSTMs). Instead of learning static temporal filters whose location/duration/resolution is shared by all videos, our LSTM based approach dynamically and adaptively adjusts its filter parameters depending on the video, by going through multiple iterations. Our proposed approach is able to function in conjunction with any per-frame or per-video-segment CNNs as well as with other types of feature representations (e.g., Fisher vectors), making it very generally applicable for many video understanding scenarios.

2 Previous works

As described in the introduction, the direction of using convolutional neural networks for video classification is becoming increasingly popular, since it allows end-to-end training of convolutional filters optimized for the training data. [14] used optical flows in addition to image feature. [8] tested

multiple different types of pooling strategies on top of per-frame CNNs, and found that the simple global max pooling of per-frame features over the entire interval performs the best for sports videos. [5] also tried multiple different (temporal) pooling strategies, gradually combining per-frame CNN responses over a short interval using their ‘slow fusion’. [15] proposed to do XYT convolution, learning space-time convolutional filters. [17] used local CNN feature maps around tracklets in videos.

However, these prior works focused only on capturing dynamics in very short video intervals without much consideration on long-term temporal structure of activity videos. Optical flow only captures differences between two consecutive frames [14]. Even with the video-based 3-D XYT CNNs [15] or trajectory CNNs [17], only the temporal dynamics within short intervals with a fixed duration (e.g., 15 frames) were captured. [12] showed a potential that considering temporal structure in terms of sub-intervals (e.g., temporal pyramid) may benefit the recognition, but they did not attempt any learning. Similarly, [7] considered multiple different temporal scales, but learning of how the system should choose such scales were not attempted. Recurrent neural networks such as LSTMs were also used to model sequences [8], but they were unable to explicitly consider different temporal sub-intervals and their structure. That is, learning to consider different intervals with different temporal resolution was not possible. [18] proposed the use of LSTM to make the system focus on different frames of videos, but it was unable to represent intervals.

The main contribution of this paper is the introduction of the temporal attention filters that allow their end-to-end training together with underlying CNN architectures. We illustrate that our temporal attention filters can be learned to focus on different temporal aspects of videos (i.e., intervals with different temporal resolutions), and experimentally confirm that such learning benefits the activity recognition. The main difference between our approach and previous temporal structure learning methods for activity recognition (e.g., [9, 11]) is that our temporal filters are designed to be fully differentiable, which allows their joint learning and testing with modern CNN architectures.

To our knowledge, this paper is the first paper to try temporal sub-interval learning using convolutional neural networks for activity recognition.

3 Recognition approach

3.1 Temporal attention filters

Each temporal attention filter learns three parameters: a center g , a stride δ and a width σ . These parameters determine where the filter is placed and the size of the interval focused on. A filter consists of N Gaussian filters separated by a stride of δ frames. The goal of this model is to learn where in the video the most useful features appear. Because the videos are of variable length, δ and g are relative to the length of the video. Based on the attention model presented in [2], we use the following equations to obtain the mean of the Gaussian filters:

$$\begin{aligned} g_n &= 0.5 \cdot T \cdot (\tilde{g}_n + 1) \\ \delta_n &= \frac{T}{N-1} \tilde{\delta}_n \\ \mu_n^i &= g_n + (i - 0.5N + 0.5)\delta_n \end{aligned} \tag{1}$$

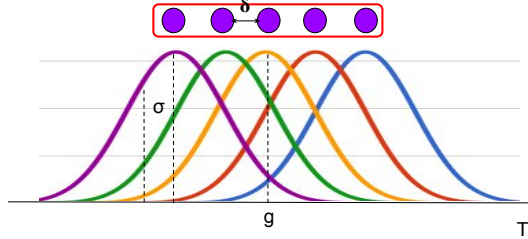


Figure 2: An illustration of our temporal attention filter. The filter is differentiable and represented with three parameters.

Using μ and σ , the N Gaussian filters are defined by:

$$F_m[i, t] = \frac{1}{Z_m} \exp\left(-\frac{(t - \mu_m^i)^2}{2\sigma_m^2}\right) \quad (2)$$

$$i \in \{0, 1, \dots, N-1\}, t \in \{0, 1, \dots, T-1\}$$

where Z_m is a normalization constant. Figure 2 shows an illustration of our temporal attention filter. If each frame has D -dimensional features, the filters, F , are applied to each dimension, taking the input of size $T \times D$ to $N \times D$ where T is the number of frames in the video. Concatenating these N features together gives a $N \times D$ -dimensional vector for any video which is used as input to a neural network for classification. Since this model is fully differentiable, all parameters can be learned using gradient descent.

Let $f_m[i, d]$ be the output of our m th temporal attention filter, given the $T \times D$ dimensional input x . Each $f_m[i, d]$ describes the response from the i th Gaussian filter on the d th elements of the input vectors. Then,

$$f_m[i, d] = F_m[i] \cdot x[:, d] = \sum_{t=0}^{T-1} F_m[i, t] \cdot x[t, d] \quad (3)$$

$$i \in \{0, 1, \dots, N-1\}, d \in \{0, 1, \dots, D-1\}$$

where $x[:, d]$ is the T -dimensional vector corresponding to the sequence of the d th element values in the underlying CNN feature vectors.

Figure 3 shows how each temporal attention filter is able to capture features from the corresponding sub-interval of the provided video with different temporal resolutions.

3.2 Recognition with temporal attention filters

As described in Figure 1, we take advantage of multiple different temporal attention filters by placing them on top of a sequence of per-frame (or per-segment) CNN models. As a result, our model is able to focus on different sub-intervals of video inputs with different temporal resolutions. Outputs of each temporal filters are concatenated and are connected to fully connected layers performing

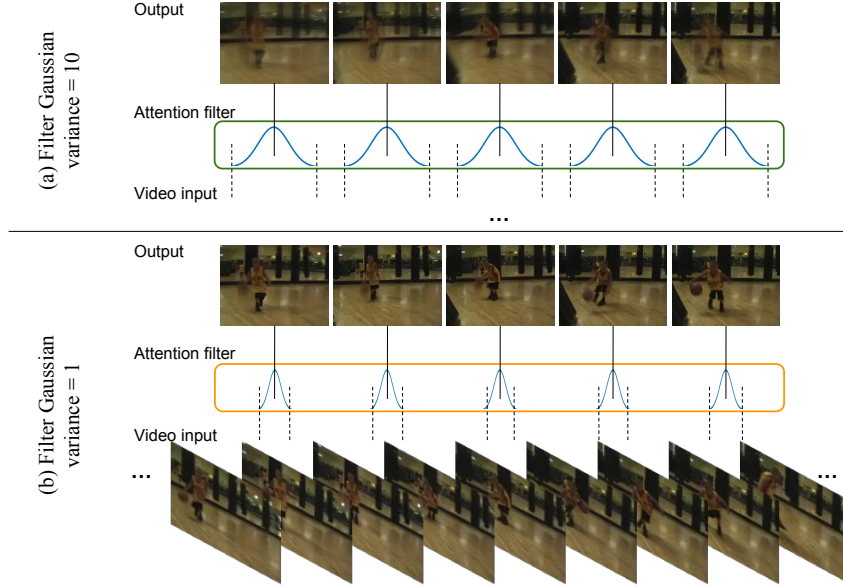


Figure 3: Examples illustrating how our temporal attention filters work once learned. They are shown with two different Gaussian filter variances. For this visualization, we used the case where raw video frames as inputs directly. In our actual implementation, the input to temporal filters are not raw video frames but CNN feature responses obtained from frames.

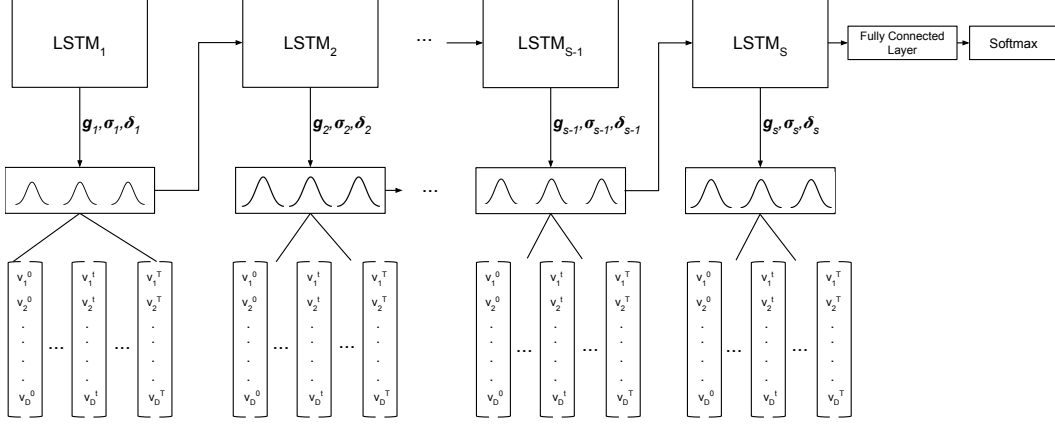


Figure 4: (a) An illustration of our temporal attention filter using an LSTM. The LSTM provides the parameters for the filters at each time step. The filters provide input to the next iteration allowing the LSTM to adjust the location of the filters. After S iterations, a fully-connected layer and softmax layer classify the video.

activity classification. If we denote the per-frame feature size as D and we have M number of temporal attention filters, each temporal filter generates the output of size $N \times D$, resulting the total dimensionality to be $M \times N \times D$. We used 2 fully connected layers (i.e., one hidden layer and one softmax layer) for the classification.

Because of the property that our temporal filters are designed to be differentiable similar to [2], we are able to backpropagate the errors through temporal attention filters reaching the underlying per-frame convolutional layers. This makes end-to-end training of the proposed model possible with video training data. Per-frame CNNs were assumed to share all parameters.

3.3 Recurrent neural networks with temporal filters

Although the model presented in Subsection 3.2 allows us to learn temporal attention filters from the training data, it was assumed that the temporal filters are static once learned and are shared across all videos. However, such assumption that relative locations of sub-events are exactly identical across all activity videos can be dangerous. A particular sub-event of the activity (e.g., a person stretching an arm in the case of ‘shake hands’) may occur earlier or faster in one video than those in the other videos, due to human action style variations. In such cases, using static temporal filters will fail to capture exact sub-events. Rather, the recognition system must learn how to dynamically and adaptively adjust locations of temporal filters depending on the video content.

Thus, in this subsection, we propose an alternative approach of using a recurrent neural network, the long short-term memory (LSTM). Figure 4 describes our overall LSTM architecture. At each iteration, our LSTM takes the entire video frames as an input and applies per-frame CNNs identical to Subsection 3.2. Next, instead of using the learned static temporal filters, previous LSTM outputs are used to decide the temporal attention filter parameters in an adaptive fashion. Our approach learns weights that models how previous LSTM iteration outputs (i.e., the abstraction of video information in the previous round) can lead to the better temporal filters in the next iteration.

More specifically, our attention filter parameters become the function of previous iteration LSTM outputs:

$$(g_t, \delta_t, \sigma_t) = W_n(h_{t-1}) = \sum_i w_i \cdot h_{t-1}(i) \quad (4)$$

where W_n is the function we need to learn modeled as a weighted sum, h_{t-1} is the LSTM hidden state vector at iteration $t - 1$ and $h_{t-1}(i)$ are its elements. These weights are initialized such that the initial iteration places g at the center of the video, and δ spans the entire duration, allowing the LSTM to get input from the entire sequence of frames.

Once more, because of the nature that our temporal attention filters are differentiable, we learn the function W_n through the backpropagation.

4 Experiments

In order to evaluate the effectiveness of the proposed fully differentiable temporal attention filters, we conducted a set of experiments comparing our approach using temporal filters against the previous conventional approaches without them while making the approaches use the same features and classifiers. These approaches were evaluated with multiple different features and multiple different datasets.

Features: We extracted VGG features [13], INRIA’s improved trajectory features (ITF) [16], and TDD features [17] which was used as input to our temporal filter model. VGG is an image-based convolutional neural network originally designed for object classification tasks. ITF and TDD are the state-of-the-art trajectory-based local video features, each taking advantage of HOG/HOF and CNN feature maps observed around of trajectories. We used the source codes of all these features provided by the authors of the corresponding papers. For TDD, we used the single-scale version of the TDD feature, since that is what was provided in their code. We used 3-frame short video segments as our unit observations. In the case of VGG features, we applied its CNN architecture to one image frame in every 3 frames, obtaining 4K-dimensional vectors from the final fully connected layer. In the case of trajectory features, we considered the trajectories ending within the 3-frame segment as inputs corresponding to the segment and took advantage of their Fisher vector representation.

Datasets: We conducted experiments with two different public video dataset: DogCentric activity dataset [3] and HMDB dataset [6]. The DogCentric dataset is a first-person video dataset, and it was chosen because that the previous pooled time series (PoT) representation [12], which illustrated potential that considering multiple sub-intervals of videos benefit the CNN-based recognition, achieved the state-of-the-art performance on it. The first-person videos (also called egocentric videos) in this dataset displays a significant amount of ego-motion (i.e., camera motion) caused by the camera wearer, and it is an extremely challenging dataset. HMDB was chosen due to its popularity. Our experiments were conducted by following each dataset’s standard evaluation setting. Both the datasets are designed for multiclass activity video classification.

Implementation and baseline classifiers: We extracted VGG features, INRIA trajectory features (ITF), and TDD features, which were used as input to our temporal filter model. In all our experiments, we used 2 hidden layers on top of our (multiple) temporal filters: one of size 2048 and one of size 10 and softmax classification.

As the basic baselines, we tested (1) max-pooling, (2) sum-pooling, and (3) mean-pooling across the time dimension, resulting in features of size $1 \times D$. These $1 \times D$ features were fed to the hidden layers for the classification, which was a standard practice as described in [4, 5, 14, 8, 15]. In addition, in order to confirm the power of our approach of ‘learning’ temporal attention filters with different location/duration/resolution, we also implemented the baselines with fixed-predetermined temporal filters (i.e., without learning). The main idea was to make the systems take advantage of temporal filters identical to ours while disabling their learning capability. This makes the baselines behave very similar to previous pooled time series method [12]. We tested (4) an approach of having a single temporal filter with fixed parameters so it would compute a weighted sum of the whole video. We then used (5) a pyramid of temporal filters (i.e., temporal pyramid) where each layer would see subintervals of the video. With 4 layers, there would be 15 filters: 1 viewing the whole video, 2 viewing half the video, 4 viewing a forth, and 8 viewing an eighth, giving a vector of size $15 \times N \times D$.

We implemented our CNN-based recognition architecture with learnable temporal filters as described in Section 3. First, we implemented our approach of learning static temporal filters shared across all video (Subsection 3.2). We tested our model’s ability to learn filters by using 15 filters (i.e., $M = 15$) with $N = 1$ or $N = 3$. Finally, as described in Subsection 3.3, we modified the model to use an LSTM to dynamically choose where to look. At each step, the temporal model took the hidden state, h from the LSTM as input, and did a linear transformation with learned weights and bias $W \cdot h + b$ to obtain g , δ and σ . We then used the same equations as above to create the filters. The LSTM ran for 4 steps and either had 1 or 3 filters (i.e., $M = 1$ or 3) with $N = 5$.

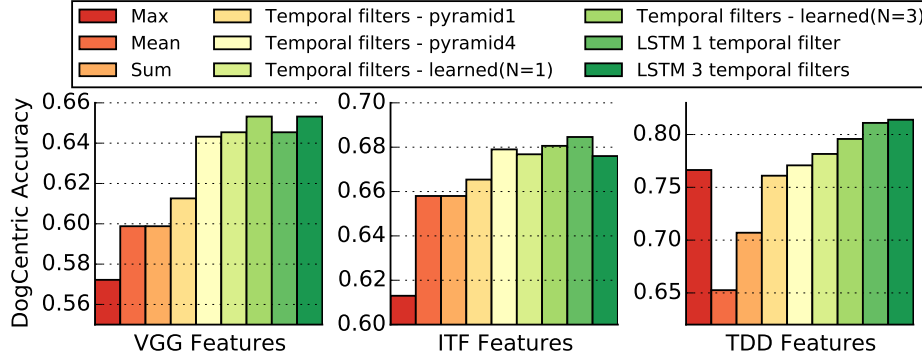


Figure 5: Classification accuracy of baselines and our proposed approach using VGG, ITF and VGG features.

Training the network To increase training data, we apply random cropping on each video frame, and randomly skip several video frames at beginning. We use log-scale of stride and variance to ensure positivity as [2]. We initialize each filter bank parameters ($\tilde{g}_m, \log \tilde{\delta}_m, \log \sigma_m^2$) with normal distribution for $\tilde{g}_m$ and 0 for $\log \tilde{\delta}_m$ and $\log \sigma_m^2$.

For all experiments, the first fully-connected layer had 4096 nodes and used a ReLU activation function. The second had either 10 nodes (for DogCentric) or 51 nodes (for HMDB) and used soft-max. The network was trained for 10000 iterations with a batch size of 100 and stochastic gradient descent with momentum set to 0.9.

4.1 DogCentric dataset

The DogCentric Activity dataset consists of 209 videos (104 training and 105 testing videos) and 10 classes. As mentioned above, it is a very challenging dataset with severe camera motion.

All of the baselines described above as well as 4 different versions of our approach (2 with static filter learning and 2 with LSTMs) were compared while using 3 different types of underlying features. Figure 5 shows the results. We are able to clearly observe that the consideration of multiple sub-intervals improves the recognition performances. The performance increased by using predetermined temporal pyramid, and our proposed approach of learning temporal filters were able to further improve the performance. Additionally, using the LSTM to dynamically choose locations gives the best performance.

The overall difference between the conventional approach of max/sum/mean pooling and our learned temporal attention filters are around 5% in VGG, 2~3% in ITF, and 4~5% in TDD. We believe our approach was more effective with VGG since it's a pure CNN-based feature making our differentiable filters to better cope with them. ITF is a completely hand-crafted feature and TDD is a partially hand-crafted feature.

Table 1 compares the performances of our approach with the previously reported state-of-the-arts. The base features we used for this table was TDD. By taking advantage of temporal filter learning and also LSTMs to dynamically adjust the filters, we were able to outperform the state-of-the-arts.

4.2 HMDB

HMDB is a relatively large-scale video dataset with 50 activity classes and more than 5000 videos.

Table 2 shows the results of our approach with the temporal attention filters compared against the approach without them. We tested this with CNN-based features, VGG and TDD this time. Unfortunately, we were unable to replicate the reported state-of-the-art recognition performance of 63.2 % recognition accuracy with the code provided by the authors. This probably is due to the difference in detailed parameter settings and the tricks to make the multi-scale TDD feature work. However, still, we were able to clearly observe that the use of temporal attention filters and

Table 1: Recognition performances of our approach on the DgoCentric dataset, compared against previously reported results of state-of-the-art approaches.

Approach	Accuracy
VGG [13]	59.9 %
Iwashita et al. [3]	60.5 %
Improved Traj. Feature (ITF) [16]	67.7 %
ITF + CNN [4]	69.2 %
Pooled time series (PoT) [12]	73.0 %
PoT + ITF	74.5 %
TDD [17]	76.6 %
Ours (temporal filters)	79.6 %
Ours (temporal filters + LSTM)	81.4 %

Table 2: A table comparing the performance of our model on the HMDB Dataset

Method	VGG Features	TDD Features
Baseline		
Max Pooling	0.3777	0.5707
Sum Pooling	0.3700	0.5577
Mean Pooling	0.3773	0.5717
Fixed Temporal Filters		
Pyramid 4	0.4156	0.5887
Learned Temporal Filters		
$N = 3$	0.425	0.5903
LSTM 3 filters	0.4303	0.5893

consideration of sub-intervals in HMDB videos increase the performance from 57 % (max temporal pooling) to 59 % (ours with temporal filters). If we can replicate the base performance of TDD reported in [17], we would be able to further increase its performance with our temporal filters based on it.

The overall trend that our temporal attention filter always benefits the recognition compared to the conventional approach of using simple temporal max/sum/mean pooling can be very clearly observed.

5 Conclusion

We introduced the concept of *temporal attention filters*. The paper presented the overall video recognition architecture taking advantage of temporal attention filters in conjunction with per-frame or per-segment CNN models, and confirmed that considering sub-intervals of activity videos using temporal filters benefits the recognition. Our temporal attention filters were designed to be differentiable, allowing the end-to-end learning of them with underlying CNN models. An approach to adaptively update the temporal filter location/duration/resolution using multiple recurrent LSTM iterations was also proposed, and its potential was experimentally confirmed.

References

- [1] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. *ACM Computing Surveys*, 43:16:1–16:43, April 2011.
- [2] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623, 2015.
- [3] Y. Iwashita, A. Takamine, R. Kurazume, and M. S. Ryoo. First-person animal activity recognition from egocentric videos. In *ICPR*, 2014.
- [4] M. Jain, J. van Gemert, and C. G. M. Snoek. University of amsterdam at THUMOS challenge 2014. In *ECCVW*, 2014.
- [5] 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.

- [6] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011.
- [7] Q. Li, Z. Qiu, T. Yao, T. Mei, Y. Rui, and J. Luo. Action recognition by learning deep multi-granular spatio-temporal video representation. In *ICMR*, 2016.
- [8] 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.
- [9] J. Niebles, C. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, 2010.
- [10] M. S. Ryoo, T. J. Fuchs, L. Xia, J. K. Aggarwal, and L. Matthies. Robot-centric activity prediction from first-person videos: What will they do to me? In *HRI*, 2015.
- [11] M. S. Ryoo and L. Matthies. First-person activity recognition: What are they doing to me? In *CVPR*, 2013.
- [12] M. S. Ryoo, B. Rothrock, and L. Matthies. Pooled motion features for first-person videos. In *CVPR*, 2015.
- [13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
- [14] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.
- [15] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015.
- [16] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.
- [17] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015.
- [18] 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.