Supplementary Materials for Temporal Gaussian Mixture Layer for Videos

AJ Piergiovanni [1] **Michael S. Ryoo** [1]

A. Implementation Details

As our base per-segment CNN, we use the I3D (Carreira & Zisserman, 2017) network pretrained on the ImageNet and Kinetics (Kay et al., 2017) datasets. I3D obtained state-of-the-art results on segmented video tasks, and this allows us to obtain reliable v_t. We also use two-stream version of InceptionV3 (Szegedy et al., 2016) pretrained on Imagenet and Kinetics as our base per-frame CNN, and compared them. We chose InceptionV3 as it is deeper than previous two-stream CNNs such as (Simonyan & Zisserman, 2014; Feichtenhofer et al., 2016). We extracted frames from the videos at 25 fps, computed TVL1 (Zach et al., 2007) optical flow, clipped to $[-20, 20]$. For InceptionV3, we computed features for every 3 frames (8 fps). For I3D, every frame was used as the input. I3D has a temporal stride of 8, resulting in 3 features per second (3 fps). By design, I3D has a temporal resolution of 99 frames, so each feature is able to capture up to 99 frames of temporal information.

We implemented our TGM layers as well as other baseline layers in PyTorch. Our default setting was as follows: for 3-layer models, we set $L = 10$ for frame-based features (i.e., InceptionV3) and $L = 5$ for segment-based features (i.e., I3D), as each segment already contains some temporal information. For 1-layer models, we set $L = 30$ for frame-based features and $L = 15$ for segment-based features. We set $M = 16$ and $C_{out} = 80$ and $C_{out} = 65$ for the last TGM layer. We found these values to work well on a held out portion of the training set of MultiTHUMOS. In all models, we used one fully-connected layer at the end to make the per-frame or per-segment classification.

We trained our models using the Adam (Kingma & Ba, 2014) optimizer with the learning rate set to 0.01. We decayed the learning rate by a factor of 10 after every 10 training epochs. We trained our models for 50 epochs. We plan to make all our source code and trained models publicly available once the paper is published.

[1]Department of Computer Science, Indiana University. Correspondence to: AJ Piergiovanni <ajpiergi@indiana.edu>, Michael Ryoo <mryoo@indiana.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

Table 1. Comparison of various values of M on MultiTHUMOS and Charades using RGB I3D features. For these experiments, 1 layer was used with $L = 15$ and $C_{out} = 16$.

	MultiTHUMOS	Charades
$M = 2$	27.8	15.5
$M = 4$	33.1	16.2
$M = 8$	34.8	17.5
$M = 16$	36.1	17.5
$M = 32$	35.7	17.1
$M = 64$	35.8	17.3

Table 2. Comparison of values of C_{out} on MultiTHUMOS and Charades using RGB I3D features. For these experiments, 1 layer was used with $L = 15$ and $M = 16$.

	MultiTHUMOS	Charades
$C_{out} = 1$	33.5	16.2
$C_{out} = 4$	34.2	17.4
$C_{out} = 8$	35.5	17.5
$C_{out} = 16$	36.1	17.5
$C_{out} = 32$	36.0	17.2
$C_{out} = 64$	36.1	17.4
$C_{out} = 80$	36.1	17.5

B. Hyperparameter Experiments

We conducted a set of experiments to compare the effects of the temporal duration, L, number of Gaussians, M, and the number of output channels, C_{out}. For these experiments, we only used the one-stream version of I3D with RGB inputs.

Effect of L: In Table 3, we compare different values of L. For these experiments, we use $M = 16$ and $C_{out} = 16$. We find that the 3-layer model with $L = 5$ performs the best. With I3D features, this allows the model to capture up to 8 seconds of information. The average activity in MultiTHUMOS is 3.3 seconds long and the maximum is 14.7 seconds long, and with this setting, the model is able to capture enough temporal context to perform well. Larger values of L capture too much temporal information, but due to the Gaussian structure, it does not drastically harm performance. Figure 3 shows that even with longer kernels, the Gaussians learn to focus mostly on the center of the interval and capture the rough duration of the activities. Thus, having too long intervals does not drastically harm performance, which

Table 3. Effect of L on MultiTHUMOS and Charades using only RGB I3D features. Note that the 3 TGM layer models have larger temporal resolution than the 1 TGM layer models for the same values of L. We also compare to using standard one-layer 1-D conv layer with different values of L.

	MultiTHUMOS			Charades		
	1 Layer	3 Layers	1-D Conv	1 Layer	3 Layers	1-D Conv
I3D Baseline	22.3	-	-	15.3	-	-
$L = 3$	30.2	31.7	26.6	15.5	16.1	15.5
$L = 5$	32.5	**37.2**	28.3	15.7	17.8	16.3
$L = 10$	34.5	35.4	31.7	16.1	18.2	16.6
$L = 15$	**36.1**	34.1	32.5	17.5	18.6	16.8
$L = 30$	32.5	33.9	26.5	18.1	**18.9**	12.1
$L = 50$	32.1	33.7	15.4	**18.3**	18.8	6.7

is in contrast to the standard 1-D convolution. Note that for Charades, the temporal kernels are learned to capture much longer temporal duration, as the average activity in charades is 12.8 seconds and larger values of L perform better.

Figure 3 illustrates examples of the learned TGM kernels of various lengths. The figure shows that the kernels focus on short temporal intervals on MultiTHUMOS even if we make the filters longer, as the activities are an average of 3.3 seconds long. On Charades, the TGM kernels learn to capture much longer intervals, as the activities are an average of 12.8 seconds long. We believe that this suggests TGMs are learning to capture information from the important necessary intervals.

In Table 3, we also report the results of using a standard 1-D conv. layer with different L values. The number of parameters in our TGM layer is independent of L, however, with the standard 1-D conv. layer, the number of parameters increases as L increases. We find that increasing L with 1-D convolution helps for small values of L, but for $L > 15$, the performance drastically drops, while TGM layers only show a small decrease.

Effect of M: In Table 1, we compare different values of M. For these experiments, we set $L = 15$ and $C_{out} = 16$. We find that $M = 16$ performs best, suggesting that smaller values of M restrict the possible temporal kernels too much. We also observe that larger values of M performs slightly worse than $M = 16$ (but not much), likely because they introduce more parameters than needed. When M and L have similar values, it allows the model to learn a sufficient number of Gaussians and create a diverse range of temporal kernels. When M is larger than L, it results in learning a kernel similar to standard 1-D convolution.

Effect of C_{out}: In Table 2, we compare different values of C_{out}. For these experiments, $L = 15$, we used 1-layer and $M = 16$. We find that C_{out} performs best when set to 16 or larger on these datasets. Larger values of C_{out} seem to capture redundant information, as it does not lower performance.

Table 4. Comparison of the different forms of temporal convolution on MultiTHUMOS using RGB I3D features. We set $L = 15$ and used 1 layer models for these experiments.

Standard 1-D Convolution (Fig. 1a)	32.5
1-D Conv with 1 Gaussian (Fig. 1b)	28.6
1-D Conv with many Gaussians (Fig. 1c)	33.2
TC-Conv with unconstrained kernel (Fig. 2)	32.8
Our TGM Layer	**36.1**

C. Comparison of Different Layer Forms

To confirm the various aspects of our design, we conducted experiments comparing different types of temporal convolution. In Fig. 1a we illustrate the standard 1-D convolution, taking $D \times T$ input and producing a $C \times T$ output, where D is the number of input channels and C is the number of output channels. In Fig. 1b, we illustrate the method of applying a Gaussian mixture kernel as 1-D convolution. Here, the Gaussian mixture kernel is shared by all D input channels and we learn a C number of such kernels. In Fig. 1c, we illustrate the approach of applying a Gaussian mixture kernel as 1-D convolution while learning D different Gaussian mixtures. This is very similar to the standard 1-D convolution, except that the filter values are constrained to have the shape of Gaussian mixtures.

Fig. 2 illustrates one more baseline. This is similar to our full TGM layer with the channel-combination (Fig. 4 in main paper). However, in this baseline, instead of learning Gaussian mixtures, we learn $C_{in} \cdot C_{out}$ number of $1 \times L$ kernels. The kernel values are left unconstrained. While the TGM layer has $2 \cdot M + C_{in} \cdot C_{out} \cdot M + C_{in} \cdot C_{out}$ parameters, this layer has $L \cdot C_{in} \cdot C_{out} \cdot M + C_{in} \cdot C_{out}$, which is more than the TGM layer.

In Table 4, we compare the results of the various above-

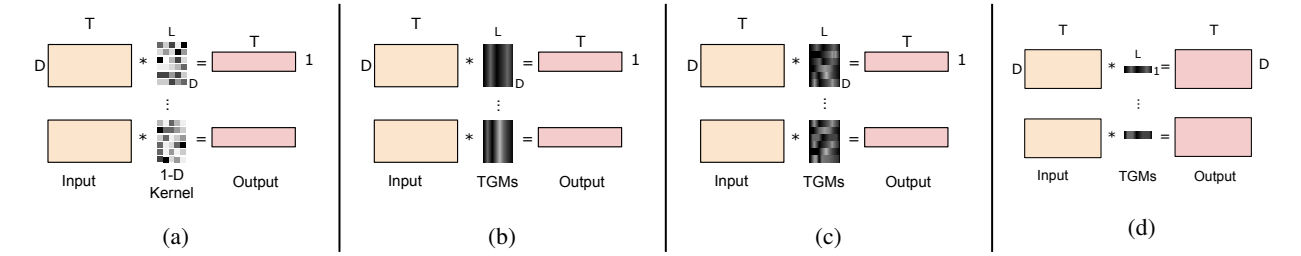


Figure 1. **(a-c)** Different forms of 1-D temporal convolutions which take a $D \times T$ input and produces a $C \times T$ output based on C number of $D \times L$ kernels: **(a)** the standard 1-D convolution, **(b)** using Gaussian mixtures for 1-D convolution while sharing Gaussian mixtures across input channels, and **(c)** using D different Gaussian mixtures for 1-D convolution. **(d)** Our TGM layer in its simplest form (i.e., 1-layer case) applying the $1 \times L$ temporal kernel in a 2-D convolutional fashion, maintaining both time and feature axis.

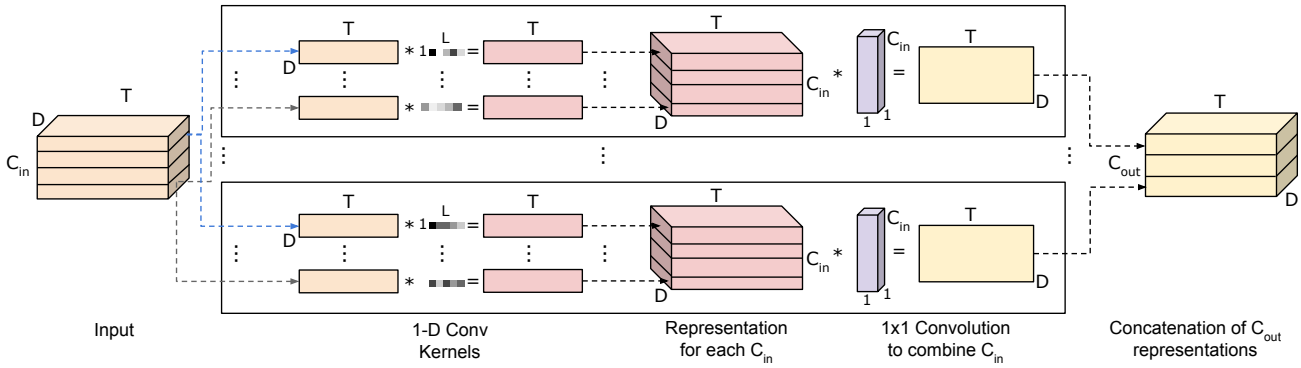


Figure 2. A temporal convolutional layer with channel combination similar to Fig. 4 (in main paper). The difference is that this layer does not learn Gaussian mixtures, but unconstrained 1-D temporal kernels.

mentioned layers on MultiTHUMOS using RGB I3D features. We find that the Fig. 1b method performs poorly, while the Fig. 1c method slightly outperforms the standard 1-D convolution. The Fig. 2 method is slightly better than the standard 1-D convolution, but performs worse than Fig. 1c. However, none of these layers perform as well as our TGM layer, confirming that both the design of learning Gaussian mixtures and maintaining temporal channel axis are important for activity detection.

D. Experiments on Additional Datasets

D.1. MLB-YouTube Dataset

D.1.1. Dataset

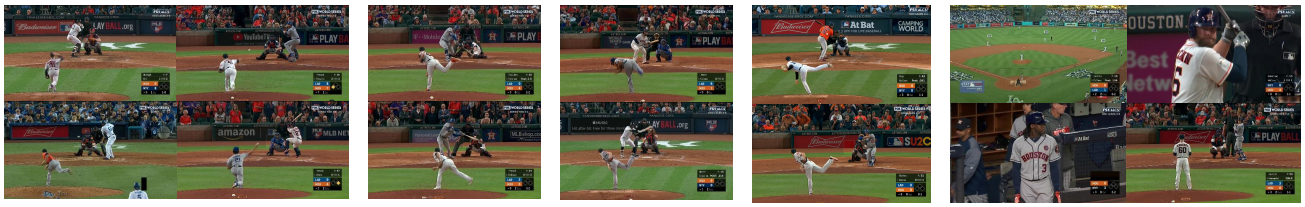
The MLB-YouTube dataset (Piergiovanni & Ryoo, 2018a) consists of 20 baseball games from the 2017 MLB postseason available on YouTube. This dataset consists of over 42 hours of video. For these experiments, we used the continuous video setting which have 2,126 1-2 minute long clips. Each clip is densely annotated with the baseball activities that occur. There are 8 activity classes: pitch, strike, ball, swing, hit, foul, hit by pitch, and bunt. Examples of some of these classes are shown in Fig. 4. Each continuous clip contains on average of 7.2 activities, giving a total of over 15,000 activity instances in the dataset.

What makes this dataset challenging is that the variation between classes is very small. In ActivityNet (Heilbron et al., 2015), for example, the difference between swimming and brushing hair is drastic. The background, motion, and even size of the person in the video is different. However, in broadcast baseball videos, the difference between a ball and a strike, or a swing and a bunt, are small. All actions are recorded from the same camera angle as we can confirm from Fig. 4.

D.1.2. Results

In Table 5, we compare various approaches on this dataset. Our TGM layers improve over the baseline by ∼6% (40.1 vs. 34.2). Additionally, we compare to methods using the super-event representation (Piergiovanni & Ryoo, 2018b), which previously achieved state-of-the-art performance on several activity detection datasets. On this dataset, our approach outperforms the super-event representation, and further the concatenation of our TGM representation with such super-event representation performs best by a significant margin (∼13% compared to the baseline). This suggests that TGMs and super-event capture different temporal information and are both useful to the detection task.

Figure 3. Illustration of several learned TGM kernels. On MultiTHUMOS, it learns to focus on shorter intervals to capture shorter events. On Charades, the Gaussians have a larger σ value, resulting in filters that attend to longer temporal durations.



(a) (b) (c) (d) (e)

Figure 4. Examples of several of the activities in the MLB-YouTube dataset: (a) Pitch, (b) Hit, (c) Bunt, (d) Hit by pitch, (e) No activity. This shows the difficulty of this dataset, as the difference between hit and bunt, swing and no swing are very small.

We further find that using multiple, standard temporal convolution layers leads to worse performance, likely due to overfitting from the large number of parameters. While using multiple TGM layers improves performance, confirming that the Gaussian structure and sparsity constraint benefits model learning.

D.2. AVA

D.2.1. Dataset

AVA (Gu et al., 2017) is a large-scale video dataset containing of 80 atomic action classes in 57k video clips. These clips are drawn from movies. Existing datasets, such as Charades, have very specific actions that depend on objects, such as holding a cup vs. holding a picture. In AVA, the actions are intentionally generic, such as sit, stand, hold, carry, etc. Further, the AVA dataset is annotated with both spatial and temporal locations of activities. Since we are interested in temporal activity detection, we follow the setting of Piergiovanni & Ryoo (2018b) and label each frame with the occurring activities while ignoring the spatial location. We evaluate performance following the same method as MultiTHUMOS, Charades and MLB-YouTube by measuring per-frame mAP.

D.2.2. Results

In Table 6, we present the results of our model. We again find that temporal convolution and LSTMs provide some benefit over the baseline, but TGM layers further improve performance. Again, combining the TGM, which captures local temporal structure, with super-events which capture global temporal structure, provides the best performance by $\sim 7.4\%$.

D.3. Context Gaiting

Context gating (Miech et al., 2017) is an layer designed to capture relationships between network activations. However, it is designed for segmented video clip classification, as it originally takes a fixed-size input. Applying it to variable length continuous videos in a sliding-window fashion is possible, and we conducted this experiment with a window

Table 5. Result mAP on the MLB-YouTube dataset using InceptionV3 and I3D to obtain features. Our TGM layers significantly outperform the baseline models.

Model	Spatial	Temporal	Two-stream
Random	13.4	13.4	13.4
InceptionV3	31.2	31.8	31.9
InceptionV3 + LSTM	32.1	33.5	34.1
InceptionV3 + 1 temporal conv	32.8	34.4	35.2
InceptionV3 + 3 temporal conv	28.4	29.8	30.1
InceptionV3 + super-events	31.5	36.2	39.6
InceptionV3 + 1 TGM	32.4	36.3	37.4
InceptionV3 + 3 TGM	33.2	38.2	38.2
InceptionV3 + 3 TGM+super-events	34.6	42.4	42.9
I3D	33.8	35.1	34.2
I3D + LSTM	36.2	37.3	39.4
I3D + 1 temporal conv	37.3	38.6	39.9
I3D + 3 temporal conv	32.4	34.6	35.6
I3D + super-events	38.7	38.6	39.1
I3D + 1 TGM	35.5	37.5	38.5
I3D + 3 TGM	36.5	38.4	40.1
I3D + 3 TGM+super-events	39.4	46.0	**47.1**

Table 6. Results on AVA dataset with the temporal annotation-only setting (i.e., frame classification without using bounding box training labels).

	mAP
Random	2.65
I3D baseline	7.5
I3D + 3 temporal conv. layers	7.9
I3D + LSTM	7.8
I3D + super-events(Piergiovanni & Ryoo, 2018b)	9.8
I3D + 1 TGMs	11.2
I3D + 3 TGMs	14.5
I3D + 3 TGMs + super-events	14.9

size of 30 (same temporal resolution as ours). When context gating is applied on top of I3D features, it gives 35.8 on MultiTHUMOS, lower than ours (44.3).

References

Carreira, J. and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Feichtenhofer, C., Pinz, A., and Zisserman, A. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1933–1941, 2016.

Gu, C., Sun, C., Vijayanarasimhan, S., Pantofaru, C., Ross, D. A., Toderici, G., Li, Y., Ricco, S., Sukthankar, R., Schmid, C., and Malik, J. AVA: A video dataset of spatio-temporally localized atomic visual actions. *arXiv preprint arXiv:1705.08421*, 2017.

Heilbron, F. C., Escorcia, V., Ghanem, B., and Niebles, J. C. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–970, 2015.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Miech, A., Laptev, I., and Sivic, J. Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, 2017.

Piergiovanni, A. and Ryoo, M. S. Fine-grained activity recognition in baseball videos. In *CVPR Workshop on Computer Vision in Sports*, 2018a.

Piergiovanni, A. and Ryoo, M. S. Learning latent super-events to detect multiple activities in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b.

Simonyan, K. and Zisserman, A. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 568–576, 2014.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

Zach, C., Pock, T., and Bischof, H. A duality based approach for realtime tv-l 1 optical flow. In *Joint Pattern Recognition Symposium*, pp. 214–223. Springer, 2007.