TEA: Temporal Excitation and Aggregation for Action Recognition

Yan Li[1] Bin Ji[2] Xintian Shi[1] Jianguo Zhang[3] Bin Kang[1] Limin Wang[2]
[1] Platform and Content Group (PCG), Tencent
[2] State Key Laboratory for Novel Software Technology, Nanjing University, China
[3] Department of Computer Science and Engineering, Southern University of Science and Technology, China
phoenixyli@tencent.com, binjinju@smail.nju.edu.cn, tinaxtshi@tencent.com
zhangjg@sustech.edu.cn, binkang@tencent.com, 07wanglimin@gmail.com

1. Temporal Convolutions in TEA

For video action recognition tasks, previous works typically adopt 3D convolutions to simultaneously model spatial and temporal features or utilize (2+1)D convolutions to decouple the temporal representation learning and the spatial feature modeling. The 3D convolutions will bring tremendous computations costs and preclude the benefits of utilizing ImageNet pre-training. Moreover, the blend of spatial and temporal modeling also makes the model harder to optimize. Thus, as shown in Figure 2 of the main text, in our proposed TEA module, we choose (2+1)D architectures and adopt separated 1D temporal convolutions to process temporal information. To train (2+1)D models, a straightforward approach is fine-tuning the 2D spatial convolutions from ImageNet pre-trained networks meanwhile initialize the parameters of 1D temporal convolutions with random noise.

However, according to our observations, it will lead to contradictions to simultaneously optimize the temporal convolutions and spatial convolutions in a unified framework because the temporal information exchange between frames brought by temporal convolutions might harm the spatial modeling ability. In this section, we will describe a useful tactic to deal with this problem for effectively optimizing temporal convolutions in video recognition models.

Before introducing the tactic, we first retrospect the recently proposed action recognition method TSM [8]. Different from previous works adopting temporal convolutions [10, 12], TSM utilizes an ingenious **shift** operator to endow the model with the temporal modeling ability without introducing any parameters into 2D CNN backbones. Concretely, given an input feature $\mathbf{X} \in \mathbb{R}^{N\times T\times C\times H\times W}$, the shift operations are denoted to shift the feature channels along the temporal dimension. Suppose the input feature is a five-dimensional tensor, the example pseudo-codes of left/right operator are as follows:

$$\begin{aligned} \mathbf{X}\left[n,t,c,h,w\right] &= \mathbf{X}\left[n,t+1,c,h,w\right], \qquad 1 \le t \le T-1 \\ \mathbf{X}\left[n,t,c,h,w\right] &= \mathbf{X}\left[n,t-1,c,h,w\right], \qquad 2 \le t \le T \end{aligned} \tag{1}$$

By utilizing such *shift* operation, the spatial features at time step t, $\mathbf{X}\left[:,t,:,:,:\right]$, achieves temporal information exchange between neighboring time steps, $t-1$ and $t+1$. In practice, the shift operation can be conducted on all or some of the feature channels, and the authors explore several shift options [8]. Finally, they find that if all or most of the feature channels are shifted, **the performance will decrease due to worse spatial modeling ability**.

The possible reason for this observation is that TSM utilizes 2D CNN backbone pre-trained on ImageNet as initializations to fine-tune the models on new video datasets. The benefit of utilizing such a pre-training approach is that the features of pre-trained ImageNet models would contain some kind of useful spatial representations. But after shifting a part of feature channels to neighboring frames, such useful spatial representations modeled by the shifted channels are no longer accessible for current frame. The newly obtained representations become the combination of three successive frames, *i.e.*, $t-1$, t and $t+1$, which are disordered and might be "meaningless" for the current frame t.

To balance these two aspects, TSM experimentally chooses to shift **a small part** of feature channels. More specifically, the first 1/8 channels are shifted left, the second 1/8 channels are shifted right, and the last 3/4 channels are fixed. Formally,

$$\begin{aligned} \mathbf{X}\left[n,t,c,h,w\right] &= \mathbf{X}\left[n,t+1,c,h,w\right], & 1 \le t \le T-1, \quad & 1 \le c \le C/8 & \textit{left shift} \\ \mathbf{X}\left[n,t,c,h,w\right] &= \mathbf{X}\left[n,t-1,c,h,w\right], & 2 \le t \le T, \quad & C/8 < c \le C/4 & \textit{right shift} \\ \mathbf{X}\left[n,t,c,h,w\right] &= \mathbf{X}\left[n,t,c,h,w\right], & 1 \le t \le T, \quad & C/4 < c \le C & \textit{unchanged} \end{aligned} \tag{2}$$

Table 1. Comparison results on Something-Something. All the models are trained with 8 input frames, and the one clip-one crop protocol is utilized for inference.

Method	Top-1 (%)	Top-5 (%)
TSM [8]	43.4	73.2
(2+1)D ResNet-Conv (Ours)	23.5	45.8
(2+1)D ResNet-CW (Ours)	43.6	73.4
(2+1)D ResNet-Shift (Ours)	**46.0**	**75.3**

This *part shift* operation has been proved effective in TSM and obtains impressive action recognition accuracies on several benchmarks.

When we think over the shift operation proposed by TSM, we find that it is actually a **special case** of general 1D temporal convolutions and we will show an example to illustrate this. Instead of conducting shift operation on input feature $\mathbf{X}$ as in Equation 2, we aim to utilize a 1D channel-wise temporal convolution to achieve the same control for $\mathbf{X}$ (we utilize channel-wise convolution for simplicity, and the formulas in Equation 3 can be extended to general convolutions.). Concretely, a 1D channel-wise temporal convolution is conducted on input features, whose kernel size is 3, the kernel weights at the shifted channels are set to fixed $[0, 0, 1]$ or $[1, 0, 0]$ and the kernel weights at unchanged channels are set to fixed $[0, 1, 0]$. Formally,

$$
\begin{array}{lccc}
\mathbf{X}_{reshape} = \mathrm{Reshape}(\mathbf{X}), & \mathbf{X} \in \mathbb{R}^{N\times T\times C\times H\times W}, & \mathbf{X}_{reshape} \in \mathbb{R}^{NHW\times C\times T} & \textit{reshape \& permute} \\
\mathbf{X}_{shift} = \mathbf{K} * \mathbf{X}_{reshape}, & \mathbf{X}_{shift} \in \mathbb{R}^{NHW\times C\times T}, & \mathbf{K} \in \mathbb{R}^{C\times 1\times 3} & \textit{temporal convolution} \\
\mathbf{K}\left[cout, 1, k\right] = 1, & 1 \le cout \le C/8, & k = 3 & \textit{left shift} \\
\mathbf{K}\left[cout, 1, k\right] = 0, & 1 \le cout \le C/8, & k = 1, 2 & \textit{left shift} \\
\mathbf{K}\left[cout, 1, k\right] = 1, & C/8 < cout \le C/4, & k = 1 & \textit{right shift} \\
\mathbf{K}\left[cout, 1, k\right] = 0, & C/8 < cout \le C/4, & k = 2, 3 & \textit{right shift} \\
\mathbf{K}\left[cout, 1, k\right] = 1, & C/4 < cout \le C, & k = 2 & \textit{unchanged} \\
\mathbf{K}\left[cout, 1, k\right] = 0, & C/4 < cout \le C, & k = 1, 3 & \textit{unchanged}
\end{array}
\tag{3}
$$

where $\mathbf{K}$ denotes convolutional kernels, and $*$ indicates the convolution operation. It's not hard to see that the Equation 3 is totally equivalent to Equation 2. The shift operation proposed in TSM can be considered as a 1D temporal convolution operation with *fixed pre-designed* kernel weights. Thus, one natural question is, *whether the performance of video action recognition can be improved by relaxing the fixed kernel weights to learnable kernel weights*. We experiment to verify this question.

The experiment is conducted based on the (2+1)D ResNet baseline. The detailed descriptions of the baseline are introduced in Section 4.3.1 of the main text. We design several variants of (2+1)D ResNet, and the only difference between these variants is the type of utilized 1D temporal convolution.

- **(2+1)D ResNet-Conv** which adopts general 1D temporal convolutions. The parameters of temporal convolutions are randomly initialized.
- **(2+1)D ResNet-CW** which utilizes channel-wise temporal convolutions. The parameters are also randomly initialized.
- **(2+1)D ResNet-Shift**. In this variant, the channel-wise temporal convolutions are also utilized, but the parameters of the temporal convolutions are initialized as in Equation 3 to perform like *part shift* operators at the beginning of the model learning.

During training, the parameters of temporal convolutions in all the three variants are learnable, and the final obtained models are evaluated on Something-Something V1 with 8 frames as input and the efficient inference protocol.

The comparison results are shown in Table 1. We first notice that when comparing the (2+1)D ResNet-Conv with (2+1)D ResNet-CW, the (2+1)D ResNet-Conv baseline fails to obtain acceptable performance. As we mentioned in the main text, different channels of spatial features capture different information; thus, the temporal combination of each channel should be different and learned independently. Moreover, the general temporal convolution will introduce lots of parameters and make the model harder to be optimized.

The second observation is that the performance of (2+1)D ResNet-CW is only slightly higher than that of TSM (43.6% *vs*. 43.4%). Although the learnable kernel weights endow the model with the ability to learn dynamic temporal information

exchange patterns, all the features channels are disarrayed with randomly initialized convolutions. It finally results in damage to the spatial feature learning capacity and counters the benefits of effective temporal representation learning.

Inspired by the *part shift* strategy utilized in TSM, (2+1)D ResNet-Shift proposes to initialize the temporal convolutions to perform as *part shift*, which grantees the spatial feature learning ability inheriting from the pre-trained ImageNet 2D CNN models. Meanwhile, along with the optimization of the models, the temporal convolutions can gradually explore more effective temporal information aggregation strategy with learnable kernel weights. Finally, this *part shift* initialization strategy obtains 46.0% top-1 accuracy, which is substantially higher than TSM.

According to the experiment, we can see that by drawing lessons from TSM and designing a *part shift* initialization strategy, the performance of action recognition can be improved by using 1D temporal convolutions with learnable kernel weights. This strategy is thus applied to each of the temporal convolutions in the proposed TEA module.

2. Training Details

In this section, we will elaborate on detailed configurations for training the TEA network on different datasets. The codes and related experimental logs will be made publicly available soon.

2.1. Model Initializations

Following the previous action recognition works [11, 8, 6], we utilize 2D CNNs pre-trained on ImageNet dataset as the initializations for our network. Notice that the proposed multiple temporal aggregation (MTA) module is based on Res2Net [1], whose architecture is different from the standard ResNet [4]. We thus select the released Res2Net50 model ($\mathrm{res2net50\_26w\_4s}$[1]) pre-trained on ImageNet to initialize the proposed network.

Although Res2Net has been proved a stronger backbone than ResNet on various image-based tasks in [1], *e.g.*, image classification, and image object detection, it will **NOT** brings many improvements for video action recognition task. As we have discussed in the ablation study section (Section 4.3.1) of the main text, the temporal modeling ability is the key factor for video-based tasks, rather than the complicated spatial representations. The experimental results in Table 1 of the main text also verify this. With more powerful 2D backbones, the action recognition performance of (2+1)D Res2Net only obtains slight improvements over (2+1)D ResNet (46.2% *vs*. 46.0%).

2.2. Hyperparameters

Most of the experimental settings are as the same as TSM [8] and STM [6]. For experiments on Kinetics and Something-Something V1 & V2, the networks are fine-tuned from ImageNet pre-trained models. All the batch normalization layers [5] are enabled during training. The learning rate and weight decay of the classification layer (a fully connected layer) are set to 5$\times$ higher than other layers. For Kinetics, the batch size, initial learning rate, weight decay, and dropout rate are set to 64, 0.01, 1e-4, and 0.5 respectively; for Something-Something, these hyperparameters are set to 64, 0.02, 5e-4 and 0.5 respectively. For these two datasets, the networks are trained for 50 epochs using stochastic gradient descent (SGD), and the learning rate is decreased by a factor of 10 at 30, 40, and 45 epochs.

When fine-tuning Kinetics models on other small datasets, *i.e.*, HMDB51 [7] and UCF101 [9], the batch normalization layers are frozen except the first one following TSN [11]. The batch size, initial learning, weight decay and dropout rate are set to 64, 0.001, 5e-4 and 0.8 for both the two datasets. The learning rate and weight decay of the classification layer are set to 5$\times$ higher than other layers. The learning rate is decreased by a factor of 10 at 10 and 20 epochs. The training procedure stops at 25 epochs.

Finally, the learning rate should match the batch size as suggested by [2]. For example, the corresponding learning rate should increase two times if the batch size scales up from 64 to 128.

3. The Effect of the Transformation Convolutions in the ME Module

When calculating feature-level motion representations in the ME module, we first apply a channel-wise transformation convolution on features at the time step $t+1$. The reason is that motions will cause spatial displacements for the same objects between two frames, and it will result in mismatched motion representation to directly compute differences between displaced features. To address this issue, we add a 3$\times$3 convolution at time step $t+1$ attempting to capture the matched regions of the same object from contexts. According to our verification, this operation leads to further improvement of TEA on Something-Something V1 (from 48.4% to 48.9%). Moreover, we found that conducting transformation on both t and $t+1$ time steps does not improve the performance but introduces more operations.

[1] https://shanghuagao.oss-cn-beijing.aliyuncs.com/res2net/res2net50_26w_4s-06e79181.pth

Table 2. Comparison results on Something-Something V1.

Method	Frame×Crops×Clips	Inference Time (ms/v)	Val Top-1 (%)
TSN (2D ResNet) [41]	8×1×1	0.0163	19.7
TSM [25]	8×1×1	0.0185	43.4
TSM [25]	16×1×1	0.0359	44.8
STM [20]	8×1×1	0.0231	47.5
I3D (3D ResNet) [43]	32×3×2	4.4642	41.6
ME (d in Figure 4)	8×1×1	0.0227	48.4
MTA (c in Figure 4)	8×1×1	0.0256	47.5
TEA	8×1×1	0.0289	**48.9**

Table 3. Comparison results on Something Something V1.

Stage	**Backbone**	**Number of the TEA Blocks**	**Frames×Crops×Clips**	**Val Top1 (%)**	**Val Top5 (%)**
conv2	ResNet50	3	8×1×1	43.5	72.2
conv3	ResNet50	4	8×1×1	45.3	74.5
conv4	ResNet50	6	8×1×1	47.1	76.2
conv5	ResNet50	3	8×1×1	46.7	75.8
TEA (conv2∼conv5)	ResNet50	16	8×1×1	**48.9**	**78.1**

4. Runtime Analysis

We show the accuracies and inference times of TEA and other methods in Table 2. All these tests are conducted on one P40 GPU, and the batch size is set to 16. The time for data loading is excluded from the evaluation. Compared with STM, TEA achieves higher accuracy with similar efficiency. Compared with TSM_{16F} and I3D, both the effectivity and efficiency of TEA_{8F} are superior. The runtime of the **2D ResNet baseline** (TSN) is nearly 1.8x faster than TEA. But its performance is **far behind** ours (19.7% *vs*. 48.9%).

We further analyze the efficiency of each component in TEA by comparing TEA with MTA and ME, respectively. We can see that the hierarchical stages in MTA cause an increase of $0.0062s$ ($\triangle t$=TEA−ME), as the multiple stages need to be sequentially processed. The increased time brought by ME is $0.0033s$ ($\triangle t$=TEA−MTA). Please note that for an input feature $\mathbf{X}$ with T timestamps, it is not required to subtract between adjacent features *timestamp-wise* and then concatenate T-1 differences. We only need to *slice* $\mathbf{X}$ along the temporal dimension *twice* to obtain the features of time 1∼T-1 and time 2∼T respectively. Then only one subtraction is performed to obtain the final feature differences. The example pseudo codes are as follows, and the time cost of this approach is only $0.0002s$.

```
# x (input features): NxTxCxHxW
f_t, __ = x.split([T-1,1], dim=1)
__, f_t1 = x.split([1,T-1], dim=1)
# diff_f (feature differences): Nx(T-1)xCxHxW
diff_f = f_t1 - f_t
```

5. The Location of the TEA Block

As described in Section 4.2 of the main text, the TEA blocks are utilized to replace all the ResNet blocks of the ResNet-50 backbone from conv2 to conv5. In this section, we conduct an ablation study to explore the different impacts caused by inserting the TEA blocks into ResNet at different locations. Specifically, we replace all the ResNet blocks with the TEA blocks at a particular stage, *e.g*., conv2, and leave all other stages, *e.g*., conv3∼conv5, unchanged. The networks are learned on the training set of Something-Something V1 and measured on its validation set. During the test, the efficient protocol (center crop×1 clip) is adopted, and the comparison results are shown in Table 3. It can be seen that, in general, the action

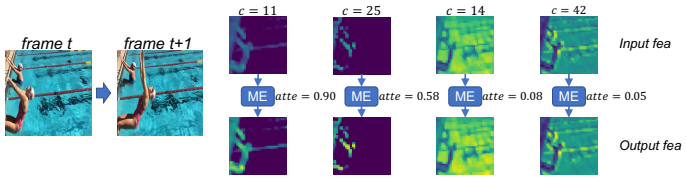

Figure 1. Visualizations of the input and output features of ME in Conv2_1 block.

Table 4. Comparison results on Something Something V2.

Method	Backbone	Frames×Crops×Clips	Val Top1 (%)	Val Top5 (%)	Test Top1 (%)	Test Top5 (%)
TSN-RGB[1]	ResNet50	16×3×2	30.0	60.5	-	-
TSM-RGB [8]	ResNet50	16×3×2	59.4	86.1	60.4	87.3
STM [6]	ResNet50	16×3×10	64.2	89.8	63.5	89.6
TEA	ResNet50	16×3×10	**65.1**	**89.9**	63.2	**89.7**

1. The results of TSN [11] on Something-Something V2 are cited from the implementation of TSM [8].

recognition performance of inserting TEA blocks into the later stages (*i.e.*, conv4/conv5, 47.1%/46.7%) is superior to that of inserting the TEA blocks into the early stages (*i.e.*, conv2/conv3, 43.5%/45.3%). The spatiotemporal features at the later stage would capture temporal information from a larger range and realize capable temporal aggregations. Thus, the TEA blocks at the later stages would have more effective and determinative impacts for improving temporal modeling ability, which finally results in higher action recognition performance. When inserting the TEA blocks into all stages of the ResNet backbone, the performance of our method further increases and achieves the best result (48.9%).

6. The verification for the assumption of ME

To verify the assumption of ME, we give a visualization example in Figure 1. We can see that different feature channels capture different information. For example, on channels 11 and 25, features model the moving swimmers, and the ME module enhances this motion information by giving a large attention weight ($\mathbf{A}$=0.90/0.58). In contrast, on channels 14 and 42, the background information is simply preserved with a quite lower attention weight, 0.08/0.05.

7. Experimental Results on Something-Something V2

In this section, we compare the proposed TEA network with other state-of-the-art methods on Something-Something V2 [3]. Something-Something V2 is a newer release version of the Something-Something dataset. It contains 168,913 training videos, 24,777 validation videos and 27,157 videos. Its size is twice larger than Something-Something V1 (108,499 videos in total). The TEA network is learned on the training set and evaluated on the validation set and test set. The accuracy inference protocol (full resolution×10 clips) is utilized for evaluation, and the results are shown in Table 4. We can see that on the validation set, our result (65.1%) outperforms those of the existing state-of-the-art methods. On the test set, the obtained number is also comparable to the state-of-the-art result (63.2% *vs*. 63.5%). These results verify the effectiveness of the proposed TEA network on Something-Something V2.

References

[1] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2net: A new multi-scale backbone architecture. *arXiv preprint arXiv:1904.01169*, 2019. 3

[2] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 3

[3] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "something something" video database for learning and evaluating visual common sense. In *ICCV*, pages 5843–5851, 2017. 5

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[6] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *ICCV*, pages 2000–2009, 2019. 3, 5

[7] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, pages 2556–2563, 2011. 3

[8] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, pages 7083–7093, 2019. 1, 2, 3, 5

[9] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 3

[10] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, pages 6450–6459, 2018. 1

[11] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, pages 20–36, 2016. 3, 5

[12] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, pages 305–321, 2018. 1