Temporal Bilinear Networks for Video Action Recognition

Yanghao Li, Sijie Song, Yuqi Li, Jiaying Liu*

Peking University

lyttonhao@gmail.com, ssj940920@pku.edu.cn, liyuqi.ne@gmail.com, liujiaying@pku.edu.cn

Abstract

Temporal modeling in videos is a fundamental yet challenging problem in computer vision. In this paper, we propose a novel Temporal Bilinear (TB) model to capture the temporal pairwise feature interactions between adjacent frames. Compared with some existing temporal methods which are limited in linear transformations, our TB model considers explicit quadratic bilinear transformations in the temporal domain for motion evolution and sequential relation modeling. We further leverage the factorized bilinear model in linear complexity and a bottleneck network design to build our TB blocks, which also constrains the parameters and computation cost. We consider two schemes in terms of the incorporation of TB blocks and the original 2D spatial convolutions, namely wide and deep Temporal Bilinear Networks (TBN). Finally, we perform experiments on several widely adopted datasets including Kinetics, UCF101 and HMDB51. The effectiveness of our TBNs is validated by comprehensive ablation analyses and comparisons with various state-of-the-art methods.

Introduction

Deep convolutional neural networks (CNNs) (Krizhevsky, Sutskever, and Hinton 2012; He et al. 2016) have witnessed the tremendous progress in computer vision over the past few years. CNNs have demonstrated their power in many visual tasks, from image classification (Russakovsky et al. 2015; He et al. 2016), object detection (Girshick et al. 2014; Ren et al. 2015), semantic segmentation (Long, Shelhamer, and Darrell 2015) and video action recognition (Simonyan and Zisserman 2014; Tran et al. 2015). However, the progress of video action recognition is relatively much slower. One of the main challenges in this area is the modeling of both spatial appearance and temporal motion across different frames. Therefore, researchers in video action recognition recently devote most efforts to the effective modeling of temporal dynamics in the deep architectures.

There are three typical schemes of temporal motion modeling for deep learning methods in action recognition. (1) Some methods capture temporal dependencies by utilizing temporal pooling (Karpathy et al. 2014) or recurrent layers (Ng et al. 2015; Karpathy et al. 2014) on the top of the 2D CNNs. However, the visual features are extracted independently by CNNs in a frame-wise manner, and the recurrent layers are fed in merely one-dimensional high-level semantic features without spatial information. It makes the temporal dynamics ignored in the preceding CNNs, especially for subtle motion dynamics. (2) Following the two-stream architecture (Simonyan and Zisserman 2014; Wang et al. 2016), many methods capture appearance and motion information separately by different stream networks with RGB and optical flow as input. Despite of the good results, it also potentially prevents the model from fully utilizing the appearance and motion information in a single network. Furthermore, the estimation of optical flow is very time- and resource-consuming. Thus, they are unsuitable for existing large-scale datasets (Kay et al. 2017). (3) Different from the above methods which use 2D CNNs, methods like C3D (Tran et al. 2015) adopt 3D convolution operators to learn spatio-temporal structures directly from RGB frames. Recently, 3D CNNs start to present their effectiveness on some large-scale action recognition datasets (Carreira and Zisserman 2017), but the temporal modeling in 3D convolution operators is still limited in linear transformation. It is unclear whether 3D convolution operators are effective enough to capture complex temporal relations across frames. In addition, 3D convolution operators also introduce much more parameters.

In this paper, we propose a new Temporal Bilinear (TB) model to enhance the capacities of CNNs to model spatio-temporal dependencies. Specifically, TB model employs bilinear transformation to capture pairwise interactions among CNN features of adjacent frames for video action recognition. We believe this explicit temporal quadratic transformations on adjacent frames are more powerful to model complex motion relations in the temporal domain. We insert the TB model into the original 2D CNNs to model appearance and motion information simultaneously. At the same time, through multiple TB blocks embedded in different layers of TBNs, multiple levels of temporal dynamics could be captured with different temporal receptive fields.

To avoid the explosion of quadratic parameters in bilinear models, we explore the factorized bilinear model (Li et al. 2017) in the temporal domain for video data. To further reduce the complexity of our TB model, we build the tempo-

ral bilinear block with a bottleneck structure, which stacks two temporal convolutional layers and one TB layer in between. The TB model could be built by existing common neural network layers, thus it could be trained effectively and seamlessly with the whole network.

To summarize, our contributions of this paper are three-fold:

- We present a novel Temporal Bilinear (TB) model to consider the temporal pairwise feature interactions across adjacent frames. By incorporating the TB model into 2D CNNs, original 2D convolution operators are complemented to capture both appearance context and motion dynamics. We also construct two different TBNs: Wide and Deep Temporal Bilinear Networks (WTBN and DTBN) to explore the effective combination of 2D convolution layers and the TB model.

- The TB model is implemented based on factorized bilinear model with linear complexity. To further reduce the computation cost, we leverage the bottleneck design to build the TB block. Thus, the complexity of our model is much lower than 3D convolution operators in terms of both parameters and computation cost.

- The effectiveness of our approach is validated on several standard benchmarks, including Kinetics (Kay et al. 2017), UCF101 (Soomro, Zamir, and Shah 2012) and HMDB51 (Kuehne et al. 2011). Our proposed method achieves superior or comparable results to state-of-the-art methods.

Related Work

Deep learning for action recognition. After the breakthrough of deep learning in image recognition (Krizhevsky, Sutskever, and Hinton 2012), many research works start to apply deep neural networks in video action recognition (Karpathy et al. 2014; Simonyan and Zisserman 2014; Ji et al. 2013; Tran et al. 2015). In (Karpathy et al. 2014), several temporal fusion strategies were explored when applying 2D CNNs on video data, but the performance is not satisfying compared to traditional hand-craft features (Wang and Schmid 2013). Later, Long Short-Term Memory (LSTM) networks appended after CNNs were investigated for better sequence modeling in action recognition (Ng et al. 2015; Donahue et al. 2015). High-level semantic features from CNNs are fed into the following recurrent layers. Thus, it is hard for the network to capture subtle motion dynamics across frames, even though LSTM and CNNs are jointly end-to-end trainable.

In (Simonyan and Zisserman 2014), the two-stream architecture was proposed for action recognition, which contains one spatial stream fed with RGB data and one temporal stream taking optical flow as input. The final results of the two-stream networks are obtained by fusion of softmax scores. Based on the two-stream structure, Temporal Segment Network (Wang et al. 2016) further performed sparse sampling and temporal fusion to capture global structures in the videos, and achieved state-of-the-art results on UCF101 (Soomro, Zamir, and Shah 2012) and HMDB51 (Kuehne et al. 2011). To obtain powerful

video-level representation with two-stream networks, Action-VLAD (Girdhar et al. 2017) incorporated learnable spatio-temporal aggregation methods in the networks. One dilemma of two-stream networks lies in the inefficient extraction of optical flow, especially for large-scale datasets (Kay et al. 2017) and practical applications.

Another typical approach for CNN-based action recognition is 3D CNNs, which extend convolution operations into the temporal domain (Ji et al. 2013). By incorporating 3D convolutional layers and 3D pooling layers, C3D (Tran et al. 2015) proposed a standard 3D CNN architecture for generic feature extraction. Based on a more powerful CNN architecture, I3D (Carreira and Zisserman 2017) inflated the 2D convolutional filters into 3D convolutions in the Inception (Ioffe and Szegedy 2015) architecture and achieved state-of-the-art results on the large-scale Kinetics dataset (Kay et al. 2017). The most obvious problem for 3D CNNs is that they inevitably bring in much more parameters. Therefore, methods like $F_{ST}CN$ (Sun et al. 2015) factorized 3D convolutional kernel with a 2D spatial kernel and a 1D temporal kernel to reduce parameters.

Instead of using optical flow as input in two-stream methods, our work directly learns spatio-temporal features from RGB frames similar to 3D CNNs. Without 3D convolutional filters, our proposed TB blocks directly capture temporal evolutions between adjacent frames. Compared with 3D convolution filters that use linear transformation across different frames, our TB model enhances the capacity of the network by modeling temporal pairwise interactions. Due to the factorization bilinear scheme and bottleneck structure design, there are much fewer parameters in our TBNs than 3D CNNs.

Bilinear models. A method called Bilinear Pooling (Lin, RoyChowdhury, and Maji 2015) was introduced to first incorporate bilinear models with CNNs for fine-grained image recognition. Bilinear Pooling calculates a global bilinear descriptor by averaging pooling of outer product of the final convolutional layer. Since the dimension of bilinear descriptors could be very large, several methods were proposed to reduce this quadratic dimensionality. Compact Bilinear Pooling (Gao et al. 2016) presented two approximation methods to obtain compact bilinear representations. Furthermore, Factorized Bilinear model (Li et al. 2017) was proposed as a generalized bilinear model which is extended to convolutional layers and meets linear complexity. In (Wang et al. 2017a), a Second-Order Response Tranform approach was propoosed to append element-wise product to a two-branch network module. Different from the above bilinear models for image recognition, we apply bilinear models on the temporal domain for video data, aiming to improve pairwise motion relations and dependency learning between adjacent frames.

Another related work is Spatiotemporal Pyramid Network (Wang et al. 2017b) for the action recognition task. This model was based on the two-stream architecture, and utilized compact bilinear to fuse high-level spatial and temporal features extracted from CNNs independently. In our work, TB blocks are introduced to model temporal relations between adjacent frames, and be embedded at different levels

of CNNs, which is more flexible. Meanwhile, our TB model are also combined with original 2D convolutional layers to jointly capture spatio-temporal structure for video action recognition.

Temporal Bilinear Networks

In this section we describe our proposed method for video action recognition. First, we discuss the existing standard temporal modeling methods. Next, we elaborate the details of our proposed Temporal Bilinear model. Finally, we introduce our design of the Temporal Bilinear block and explain how we incorporate it into the current 2D CNNs.

Temporal Modeling Methods

Suppose we have the input features (usually the filter responses of one layer in the network) of T frames. For simplicity, here we assume the features from each time step is one-dimensional. And we denote them as $\{\mathbf{x}^1, ..., \mathbf{x}^i, ..., \mathbf{x}^T\}$ where $\mathbf{x}^i \in \mathbb{R}^C$ and C is the feature dimension, we aim to aggregate these features in the temporal domain for temporal modeling. The output signals of the temporal modeling are defined as $\{\mathbf{y}^1, ..., \mathbf{y}^i, ..., \mathbf{y}^{T'}\}$ where $\mathbf{y}^i \in \mathbb{R}^C$ and T' is the output temporal dimension. In practice, each output feature corresponds to several consecutive input frames. In the following, we consider one output signal $\mathbf{y}$ and its corresponding input features which are centered at $\mathbf{x}^i$. Then the temporal modeling methods could be formulated as follows:

$$
\begin{aligned}
\mathbf{y} &= g(\{\mathbf{x}^{i+j}, l(k) \le j \le r(k)\}), \\
l(k) &= 1 - \lfloor (k+1)/2 \rfloor, \quad r(k) = \lfloor k/2 \rfloor,
\end{aligned}
\tag{1}
$$

where k is the number of considered input frames (*e.g.* the kernel size in the pooling and convolutional layers) and $g(\cdot)$ is the aggregation function. Next we discuss two standard temporal modeling methods for the aggregation function g.

Temporal Pooling.   A natural choice of aggregation function g is temporal pooling (Ng et al. 2015; Tran et al. 2015), which extends the traditional spatial pooling layers to temporal domain. The common pooling strategies could be max or average pooling:

$$
y_c = \max_{j=l(k)}^{r(k)} x_c^{i+j}, \quad y_c = \frac{1}{k} \sum_{j=l(k)}^{r(k)} x_c^{i+j},
\tag{2}
$$

where y_c and x_c^{i+j} are the c-th element of $\mathbf{y}$ and $\mathbf{x}^{i+j}$, respectively. Such pooling operations are easy to implement and fast to compute, but they ignore valuable implicit relations between different frames.

Temporal Convolution.   Similar to temporal pooling, the temporal convolution (Tran et al. 2015; Carreira and Zisserman 2017) is extended from the spatial convolution operator. It performs learnable transformation on input frames as follows:

$$
y_c = \sum_{j=l(k)}^{r(k)} \mathbf{W}_c^j \mathbf{x}^{i+j},
\tag{3}
$$

where $\mathbf{W}_c^j$ is the weight matrix for the c-th output neuron. Temporal convolution learns the transformation within several adjacent frames. However, the expressiveness of such linear transformation is limited to model complex motion structures.

These temporal modeling methods could also be combined with spatial domain operators and extended to 3D pooling or convolutional layers (Tran et al. 2015; Carreira and Zisserman 2017).

Temporal Bilinear Model

The above existing temporal modeling methods are lack of the capacity to explicitly capture interactions between adjacent frames. We are motivated to exploit bilinear transformations by a novel Temporal Bilinear model for video action recognition.

Formulation.   Following the bilinear models in image recognition (Lin, RoyChowdhury, and Maji 2015; Li et al. 2017), we define a generic temporal bilinear operation in deep neural networks as:

$$
y_c = \mathbf{x}^{i^T} \mathbf{W}_c \mathbf{x}^{i+1},
\tag{4}
$$

where $\mathbf{W}_c \in \mathbb{R}^{C \times C}$ is the interaction weight matrix between the two adjacent frames. Since each time we only consider one output neuron c, we omit this subscript for simplicity.

Although the above bilinear model is capable of capturing the temporal interactions, it introduces a quadratic number of parameters in the weight matrix $\mathbf{W}$. Following the factorization scheme in (Li et al. 2017), we adopt a factorized bilinear weight to reduce the computation cost and parameter complexity as follows:

$$
y = \mathbf{x}^{i^T} \mathbf{F}^T \mathbf{F} \mathbf{x}^{i+1},
\tag{5}
$$

where $\mathbf{F} \in \mathbb{R}^{p \times C}$ is the factorized interaction weight between the i-th and $i+1$-th input frames with $p \in \mathbb{N}_0^+$ factors. The factor number p constrains the complexity of the TB model. To explain the TB model more clearly, Eq. (5) can be expanded as:

$$
y = \sum_{j=1}^{C} \sum_{k=1}^{C} \langle \mathbf{f}_{\cdot j}, \mathbf{f}_{\cdot k} \rangle x_j^i x_k^{i+1},
\tag{6}
$$

where x_j^i and x_k^{i+1} correspond to the j-th and k-th variables of the input features $\mathbf{x}^i$ and $\mathbf{x}^{i+1}$, $\mathbf{f}_{\cdot j}$ is the j-th column of $\mathbf{F}$ and $\langle \mathbf{f}_{\cdot j}, \mathbf{f}_{\cdot k} \rangle$ calculates the inner product of $\mathbf{f}_{\cdot j}$ and $\mathbf{f}_{\cdot k}$. Therefore, each pair of the variables between two adjacent frames has their own explicit interaction weight. Meanwhile, the shared p factors also reduce the risk of overfitting in the bilinear model.

Instantiation.   For video action recognition, the input feature $\mathbf{x}^i$ is usually three dimensional. To implement the above TB model in Eq. (5) efficiently, we propose a general TB module for 3D feature maps using existing common neural network operators. Suppose the input feature map $\mathbf{x}$ is
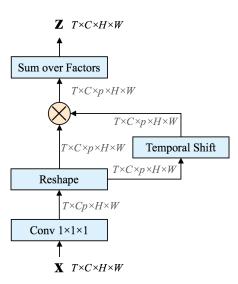
Figure 1: The structure of Temporal Bilinear module. The feature maps are shown as the shape of their tensors. "$\otimes$" denotes element-wise multiplication.

Figure 2: Two Temporal Bilinear Blocks. (a) Using one TB module. (b) Using Bottleneck structure.

$T \times C \times H \times W$ where C is the number of feature channel, H and W are the height and width of the feature map. Figure 1 shows an example of the TB module. Here the output temporal dimension is the same as the input, *i.e.*, $T' = T$.

The TB module is based on Eq. (5) and extended to combine with spatial domain. First, the convolution operator with Cp filters calculates the transformation $\mathbf{F}\mathbf{x}^i$. Then we use a temporal shift operator, which could be implemented by some indexing operators, to create the tensor $\mathbf{F}\mathbf{x}^{i+1}$ where the time index starts from 2 (The last element is padded with the T-th frame). Finally, we utilize the element-wise multiplication and sum over the factor axis (the third axis) to obtain the final TB results $\mathbf{x}^{i^T} \mathbf{F}^T \mathbf{F} \mathbf{x}^{i+1}$ for each spatial and temporal element. Therefore, the TB model could be easily implemented and incorporated into the standard CNN architectures. Note that different from the factorization implementation in (Li et al. 2017), our TB module is built on standard neural network operators in common deep learning platforms. Thus, it could be fully optimized by the standard optimization libraries such as cuDNN (Chetlur et al. 2014).

Temporal Bilinear Block. Although the complexity of TB model in Eq. (5) is linear with the factor number p and feature dimension C, it is still p times larger than 1×1 convolution operators. Following the bottleneck design of (He et al. 2016), we build a bottleneck Temporal Bilinear block to further reduce computation as shown in Figure 2. Since our TB block focuses on temporal modeling, we append two temporal convolution operators before and after the TB module. The number of output channels of the first temporal convolution and the TB module are set as $C/4$. This reduces the computation of the TB module by 1/16. Table 1 compares the complexity and temporal Receptive Field Size (RFS) of the proposed TB blocks and 2D/3D convolution operators. As indicated, our bottleneck TB block reduces the complexity of
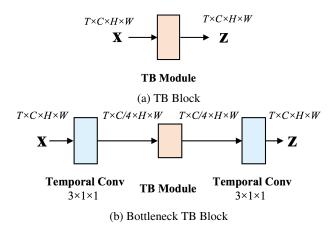
parameters and computation by nearly 1/3, even lower than 3×3 2D convolution operator. Further, the bottleneck TB block also achieves larger temporal RFS due to the stacked combination of temporal convolutions.

Temporal Bilinear Networks

Our TB blocks are flexible to incorporate with standard CNNs. In this paper we adopt ResNet (He et al. 2016) owing to its good performance and simplicity. We first introduce the 2D ResNet baselines and then describe our proposed TBNs.

2D CNN Baseline. In this paper, we adopt 2D ResNet and C3D ResNet as our baseline CNN structures. Table 2 shows the 2D ResNet-18 and C3D ResNet-18 structures. The input video clip consists of 8 frames with the resolution of 112×112. Note that the 2D kernels in ResNet-18 are equivalent to $1 \times k \times k$ kernels.

Wide and Deep TBNs. Our TB blocks are flexible to insert into both C2D and C3D ResNet. Since the TB blocks are responsible for temporal modeling, we mainly focus on incorporating them into C2D ResNet and compare the full model with C3D ResNet to validate its effectiveness in temporal domain. This also reduces the complexity of parameters and computation with C2D ResNet. To combine with spatial convolution operators in the residual block, we investigate parallel and serial integration schemes.

Figure 3 shows the structures of the proposed Wide and Deep TB blocks in terms of the integration scheme. Note that we also add the identify path of the 3×3 Conv to keep the original appearance stream. Therefore, for the Wide TB block, the appearance and temporal information is learned in two parallel paths. While in the Deep TB block, temporal modeling is appended after the spatial convolutions. Finally, we replace the original block (Figure 3a) in ResNet with our proposed TB blocks to construct Wide or Deep TBNs.

Method	Parameter	Computation	Temporal RFS
2D Conv (3×3)	$9C^2$	$O(9QC^2)$	1
3D Conv ($3 \times 3 \times 3$)	$27C^2$	$O(27QC^2)$	3
TB Block	$pC^2 = 20C^2$	$O(pQC^2) = O(20QC^2)$	2
Bottleneck TB Block	$(6 + \frac{p}{16})C^2 = 7.25C^2$	$O((6 + \frac{p}{16})QC^2) = O(7.25QC^2)$	6

Table 1: The comparisons with the proposed TB blocks and convolution operators. The value of Q depends on the strides of spatial and temporal operators, $e.g.$, $Q = HWT$ when stride is 1. The factor number p is set as 20 (Li et al. 2017). Temporal RFS represents temporal Receptive Field Size.

layer name	C2D ResNet-18		C3D ResNet-18	
	layers	output	layers	output
conv1	$7 \times 7, 64$, stride 2×2	$8 \times 56 \times 56$	$3 \times 7 \times 7, 64$, stride $1 \times 2 \times 2$	$8 \times 56 \times 56$
res1	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$8 \times 56 \times 56$	$\begin{bmatrix} 3 \times 3 \times 3, 64 \\ 3 \times 3 \times 3, 64 \end{bmatrix} \times 2$	$8 \times 56 \times 56$
res2	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$8 \times 28 \times 28$	$\begin{bmatrix} 3 \times 3 \times 3, 128 \\ 3 \times 3 \times 3, 128 \end{bmatrix} \times 2$	$4 \times 28 \times 28$
res3	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$8 \times 14 \times 14$	$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix} \times 2$	$2 \times 14 \times 14$
res4	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$8 \times 7 \times 7$	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix} \times 2$	$1 \times 7 \times 7$
	global average pooling, fc	$1 \times 1 \times 1$	global average pooling, fc	$1 \times 1 \times 1$

Table 2: The architectures for C2D and C3D ResNet-18. The dimension of feature maps is $T \times H \times W$. The input size is $8 \times 112 \times 112$. Residual blocks are shown in brackets and followed by the repeated numbers of blocks.
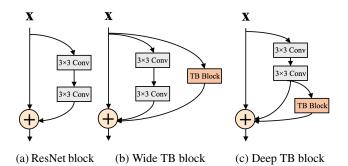

(a) ResNet block   (b) Wide TB block   (c) Deep TB block

Figure 3: The structures of the original ResNet block, proposed Wide TB block and Deep TB block.

Experiments

In this section, we first conduct comprehensive ablation studies on Mini-Kinectis-200 dataset (Xie et al. 2017). Then we compare the results with state-of-the-art methods on Kinetics (Kay et al. 2017), UCF101 (Soomro, Zamir, and Shah 2012) and HMDB51 (Kuehne et al. 2011) datasets. The proposed Wide and Deep TBNs are denoted by WTBN and DTBN, respectively.

Implementation Details. Our models are trained on the training set of Kinetics dataset from scratch. All the network weights are initialized by the method in (He et al. 2015). Following (Tran et al. 2017), the networks take $8 \times 112 \times 112$ clips as input. The frame sampling stride is set as 4. The video frames are scaled to 128×170 and randomly cropped to 112×112. We train our models for 150 epochs with an initial learning rate of 0.1, which is decayed by a factor of 10 after 45, 90, 125 epochs. We use SGD as the optimizer with a weight decay of 0.0005 and batch size of 384. The standard augmentation methods like random cropping and random flipping are adopted during training for all the methods. For TBNs, we set the factor number p as 20 and also adopt the Dropfactor scheme (Li et al. 2017) to mitigate overfitting.

For testing, following the common evaluation scheme (Wang et al. 2016; Tran et al. 2017), we uniformly sample 15 clips from input videos and then generate 10 crops for each clip. The final prediction results are obtained by averaging scores of all the clips.

Ablation Study

In this section, we investigate the design of TBNs with different ablation experiments. Since the full Kinetics dataset is quite large, we adopt the Mini-Kinetics-200 (Xie et al. 2017) for evaluation to speed up. It consists of 200 categories with most training examples from Kinetics. There are 80k and 5k videos in training and validation sets. For the baselines and TBNs, we utilize the ResNet-18 as our default backbone.

Stage to embed TB blocks. Table 3a compares the results of ResNet-18 with TB blocks embedded in different stages. We replace the two ResNet blocks in one stage with our Wide or Deep TB blocks. We can see that each TBN can lead to significant improvements (around 3% to 5%) on the C2D

Model	Stage	Top-1	Top-5
C2D	-	64.3	86.7
WTBN	res_1	68.3	89.4
WTBN	res_2	69.0	**89.8**
WTBN	res_3	**69.2**	89.4
WTBN	res_4	67.9	88.5
DTBN	res_1	**68.8**	89.2
DTBN	res_2	68.6	89.3
DTBN	res_3	**68.8**	**89.4**
DTBN	res_4	66.9	88.5

(a) Stages

Model	Blocks	Top-1	Top-5
C2D	0	64.3	86.7
WTBN	2	67.9	88.5
WTBN	4	68.8	89.1
WTBN	6	**69.5**	**89.4**
DTBN	2	66.9	88.5
DTBN	4	68.3	89.4
DTBN	6	**69.0**	**89.6**

(b) TB Blocks

Model	Params	Top-1	Top-5
C2D	11.3M	64.3	86.7
w/o bottleneck	21.1M	66.6	87.8
w bottleneck	12.9M	**68.8**	**89.1**
C3D	33.3M	66.2	87.6
WTBN C2D	11.4M	**69.0**	**89.8**
WTBN C3D	33.4M	67.2	88.3

(c) Bottleneck Design & C3D TBN

Table 3: Ablations on Mini-Kinetics-200. The default backbone is ResNet-18. Evaluated by Top-1 and Top-5 accuracy (%).

Method	Backbone	Top-1	Top-5
CNN+LSTM (Kay et al. 2017)	ResNet-50	57.0	79.0
RGB-Stream (Kay et al. 2017)	ResNet-50	56.0	77.3
C3D (Kay et al. 2017)	VGG-11	56.1	79.5
I3D-RGB (Carreira and Zisserman 2017)	Inception	68.4	88.0
3D-Res (Hara, Kataoka, and Satoh 2017)	ResNet-34	58.0	81.3
TSN-RGB (Wang et al. 2016)	Inception	69.1	88.7
C2D/TBN	ResNet-18	61.1→65.0	83.7→86.4
C2D/TBN	ResNet-34	65.4→69.5	86.4→88.9
C2D/TBN	ResNet-50	66.9→**70.1**	87.2→**89.3**

Table 4: Comparisons on the validation set of Kinetics. Note that the first four methods are evaluated on testing set. Here we only compare the methods that use only RGB as input and are trained from scratch.

baseline for both WTBN and DTBN, which demonstrates the effectiveness of our TB model compared to the simple temporal pooling in C2D. The improvement of TB blocks on res_4 is relatively smaller. It is probably because res_4 is more related to high-level semantic features with insufficient temporal motion information.

Number of TB blocks. We also investigate adding more TB blocks in TBN as shown in Table 3b. We add 2 (res_4), 4 (res_4, res_3) and 6 (res_4, res_3, res_2) TB blocks in WTBN and DTBN, respectively. Table 3b shows that more TB blocks lead to better results in general, which validates the capacity of temporal modeling of TBNs. It is also demonstrated that multiple TB blocks with larger temporal receptive fields perform better in modeling long-term temporal dependencies.

Bottleneck design. To validate the effectiveness of our bottleneck structure design in Figure 2, we compare the results of TBNs with and without the bottleneck at the top of Table 3c. The two TBNs both use 4 TB blocks. The results demonstrate the bottleneck structure not only improves the performance (by 2.2%) but also reduces the number of parameters a lot (by 39%). Note that compared to C2D baselines, our bottleneck TBN achieves a significant improvement in performance with only 14% additional parameters.

Combined with C3D ResNet. In the above comparisons, the backbone network of TBNs is C2D ResNet. We further study the performance of adding TB blocks into C3D ResNet. The bottom of Table 3c shows the results of adding 2 TB blocks (res_2) into C2D and C3D, respectively. From the results, we can see that our WTBN C3D improves 1% on C3D with almost equal number of parameters. It demonstrates that our TB blocks capture complementary information with 3D temporal convolutions. WTBN C3D does not achieve higher performance than WTBN C2D. It is mainly because of the gradual decrease of the temporal dimension in C3D, as shown in Table 2, which weakens the capacity of temporal modeling in TB blocks.

Evaluation on Multiple Datasets

In this section, we compare our TBNs with other methods on multiple datasets, including Kinetics, UCF101 and HMDB51. Since the performance of WTBN and DTBN is slightly different, here we adopt WTBN with ResNet-18, ResNet-34 and ResNet-50 as our backbone networks. We add 6 TB blocks (in res_2, res_3 and res_4) for ResNet-18, 5 TB blocks (2 in res_2 and 3 in res_3) for ResNet-34, and 2 TB blocks (2 in res_3) For ResNet-50 in WTBN.

Results on Full Kinetics Dataset Kinetics (Kay et al. 2017) is a large-scale video action recognition dataset, which contains around 240k training videos and 20k validation videos with 400 action classes.

Table 4 shows the results compared to some state-of-the-art methods. For a fair comparison, we consider the methods that only use RGB as input and are trained from scratch. Our

Method	Pretrain	backbone	UCF101	HMDB51
Two-Stream-RGB (Simonyan and Zisserman 2014)	ImageNet	VGG-M	73.0	40.5
TDD Spatial (Wang, Qiao, and Tang 2015)	ImageNet	VGG-M	82.8	50.0
Res-RGB (Feichtenhofer, Pinz, and Wildes 2016)	ImageNet	ResNet-50	82.3	43.4
TSN-RGB (Wang et al. 2016)	ImageNet	Inception	**85.1**	**51.0**
I3D-RGB (Kay et al. 2017)	ImageNet	Inception	84.5	49.8
C2D	ImageNet	ResNet-18	76.9	41.2
TBN	ImageNet	ResNet-18	77.8	42.7
TBN	ImageNet	ResNet-34	**81.4**	**46.4**
C3D (Tran et al. 2015)	Sports-1M	VGG-11	82.3	51.6
TSN-RGB (Wang et al. 2016)	ImageNet+Kinetics	Inception	91.1	-
I3D-RGB (Carreira and Zisserman 2017)	ImageNet+Kinetics	Inception	**95.6**	**74.8**
C2D	Kinetics	ResNet-18	85.0	53.9
TBN	Kinetics	ResNet-18	89.6	62.2
TBN	Kinetics	ResNet-34	**93.6**	**69.4**

Table 5: Comparisons with state-of-the-art methods on UCF101 and HMDB51. The Top-1 accuracy (%) is reported over 3 splits. Note we only consider methods that use only RGB as input.

proposed TBNs significantly improve the baseline methods. Meantime, our TBN with ResNet-18 achieves comparable results with C2D using ResNet-34 as backbone which has nearly twice the number of parameters. It demonstrates that the improvement of TBN is not just increasing depth and it is complementary to using deeper network. Compared to the recent state-of-the-art I3D (Carreira and Zisserman 2017) and TSN (Wang et al. 2016),our method achieves the best Top-1 accuracy (70.1%). Note that current published state-of-the-art methods could achieve higher performance, like (Bian et al. 2017), by utilizing more modalities, larger spatial resolutions and deeper structures.

Results on UCF101 and HMDB51 Datasets   We transfer the learned TBN models to two widely adopted action recognition datasets: UCF101 (Soomro, Zamir, and Shah 2012) and HMDB51 (Kuehne et al. 2011). UCF101 contains around 13,320 videos with 101 action classes, while HMDB51 has 6,766 videos from 51 action categories. We use the models trained on Kinetics or ImageNet as initialization and report the averaged accuracy over three splits. For finetuning, we use the same settings as Kinetics, but change the learning rate to 0.001 with total 100 training epochs.

The results are summarized in Table 5. Our TBN consistently outperforms the baseline C2D method, regardless of the employed datasets in pretraining. It is observed that better results are obtained with ResNet-34. And the performance is further improved when pretrained with Kinetics (*e.g.*, from 81.4% to 93.6% on UCF101 for TBN with ResNet-34), owing to its large-scale and high-quality video data. Finally, our TBN obtains comparable performance with other state-of-the-art methods like TSN (Wang et al. 2016) and I3D (Carreira and Zisserman 2017) which adopt deeper network structures pretrained on ImageNet and Kinetics.

Conclusions

In this paper, we have presented the Temporal Bilinear (TB) model to incorporate temporal pairwise interactions in neural networks. The factorized bilinear model and the bottleneck design bring fewer parameters and lower computational complexity. Besides, our TB block is very compact and flexible to combine with existing 2D or 3D CNNs. The Temporal Bilinear Networks (TBN) achieve consistent improvements over baselines in several video action recognition benchmarks. We believe that TB model can be an essential component for temporal modeling and we will make efforts to apply TBNs to other video domain tasks in the future.

References

Bian, Y.; Gan, C.; Liu, X.; Li, F.; Long, X.; Li, Y.; Qi, H.; Zhou, J.; Wen, S.; and Lin, Y. 2017. Revisiting the effectiveness of off-the-shelf temporal modeling approaches for large-scale video classification. *arXiv preprint arXiv:1708. 03805*.

Carreira, J., and Zisserman, A. 2017. Quo vadis, action recognition? a new model and the Kinetics dataset. In *CVPR*.

Chetlur, S.; Woolley, C.; Vandermersch, P.; Cohen, J.; Tran, J.; Catanzaro, B.; and Shelhamer, E. 2014. cuDNN: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*.

Donahue, J.; Anne Hendricks, L.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; and Darrell, T. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*.

Feichtenhofer, C.; Pinz, A.; and Wildes, R. 2016. Spatiotemporal residual networks for video action recognition. In *NIPS*.

Gao, Y.; Beijbom, O.; Zhang, N.; and Darrell, T. 2016. Compact bilinear pooling. In *CVPR*.

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

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

Hara, K.; Kataoka, H.; and Satoh, Y. 2017. Learning spatio-temporal features with 3D residual networks for action recognition. In *ICCV Workshop*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *ICML*.

Ji, S.; Xu, W.; Yang, M.; and Yu, K. 2013. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(1):221–231.

Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; and Fei-Fei, L. 2014. Large-scale video classification with convolutional neural networks. In *CVPR*.

Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; Suleyman, M.; and Zisserman, A. 2017. The Kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet classification with deep convolutional neural networks. In *NIPS*.

Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; and Serre, T. 2011. HMDB: a large video database for human motion recognition. In *ICCV*.

Li, Y.; Wang, N.; Liu, J.; and Hou, X. 2017. Factorized bilinear models for image recognition. In *ICCV*.

Lin, T.-Y.; RoyChowdhury, A.; and Maji, S. 2015. Bilinear CNN models for fine-grained visual recognition. In *CVPR*.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *CVPR*.

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

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision* 115(3):211–252.

Simonyan, K., and Zisserman, A. 2014. Two-stream convolutional networks for action recognition in videos. In *NIPS*.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Sun, L.; Jia, K.; Yeung, D.-Y.; and Shi, B. E. 2015. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*.

Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; and Paluri, M. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*.

Tran, D.; Ray, J.; Shou, Z.; Chang, S.-F.; and Paluri, M. 2017. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint arXiv:1708.05038*.

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

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Van Gool, L. 2016. Temporal segment networks: towards good practices for deep action recognition. In *ECCV*.

Wang, Y.; Xie, L.; Liu, C.; Qiao, S.; Zhang, Y.; Zhang, W.; Tian, Q.; and Yuille, A. L. 2017a. Sort: Second-order response transform for visual recognition. In *ICCV*, 1368–1377.

Wang, Y.; Long, M.; Wang, J.; and Yu, P. S. 2017b. Spatiotemporal pyramid network for video action recognition. In *CVPR*.

Wang, L.; Qiao, Y.; and Tang, X. 2015. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*.

Xie, S.; Sun, C.; Huang, J.; Tu, Z.; and Murphy, K. 2017. Rethinking spatiotemporal feature learning for video understanding. *arXiv preprint arXiv:1712.04851*.