Temporal Shift Module for Efficient Video Understanding

Ji Lin
MIT
jilin@mit.edu

Chuang Gan
MIT-IBM Watson AI Lab
ganchuang1990@gmail.com

Song Han
MIT
songhan@mit.edu

Abstract

The explosive growth in online video streaming gives rise to challenges on efficiently extracting the spatial-temporal information to perform video understanding. Conventional 2D CNNs are computationally cheap but cannot capture long-term temporal relationships; 3D CNN based methods can achieve good performance but are computationally intensive, making it expensive to deploy. In this paper, we propose a generic and effective Temporal Shift Module (TSM) that enjoys both high efficiency and high performance. Specifically, it can achieve the performance of 3D CNN but maintain 2D complexity. The central idea of TSM is to shift part of the channels along the temporal dimension, which facilitates information exchange among neighboring frames. TSM can be inserted into 2D CNNs to achieve temporal modeling at the cost of zero FLOPs and zero parameters. On the Something-Something-V1 dataset which focuses on temporal modeling, we achieved better results than I3D family and ECO family using $6\times$ and $2.7\times$ fewer FLOPs respectively. Measured on P100 GPU, our single model achieved 1.8% higher accuracy at $8\times$ lower latency and $12\times$ higher throughput compared to I3D. Remarkably, our framework ranks the first on both Something-Something V1 and V2 leaderboards upon this paper's submission.

1. Introduction

Computation-efficient video understanding is important for real-world deployment, both on the cloud and on the edge. For example, YouTube has 10^5 hours of videos uploaded every day to be processed for recommendation and ads ranking; tera-bytes of sensitive videos in hospitals need to be processed locally on edge devices to protect privacy. All these industry applications require both accurate and efficient video understanding.

Deep learning has become the standard for video understanding over the years [39, 42, 3, 43, 53, 46, 51]. One key difference between video recognition and image recognition is the need for *temporal modeling*. For example, to distinguish between opening and closing a box, reversing the order will give opposite results, so temporal modeling is critical. A straightforward approach for video understanding is to directly use 2D CNN [20, 33, 42]. However, 2D CNN on individual frames cannot model temporal information. 3D CNNs [39, 3] can jointly learn spatial and temporal features, while the computation cost for 3D CNN is large, making the production deployment expensive. There have been works to trade off between temporal modeling ability and computation, such as post-hoc fusion [11, 7, 51, 5] and mid-level temporal fusion [53, 46, 40]. Such methods sacrifice low-level temporal modeling for efficiency, but much of the useful information is lost during the feature extraction before the temporal fusion happens.

In this paper, we address the dilemma by proposing a *zero-FLOP, zero-parameter* Temporal Shift Module (TSM) that can be inserted into any 2D CNN to enable temporal modeling. For an activation in video model $A \in \mathbb{R}^{N\times C\times T\times H\times W}$,

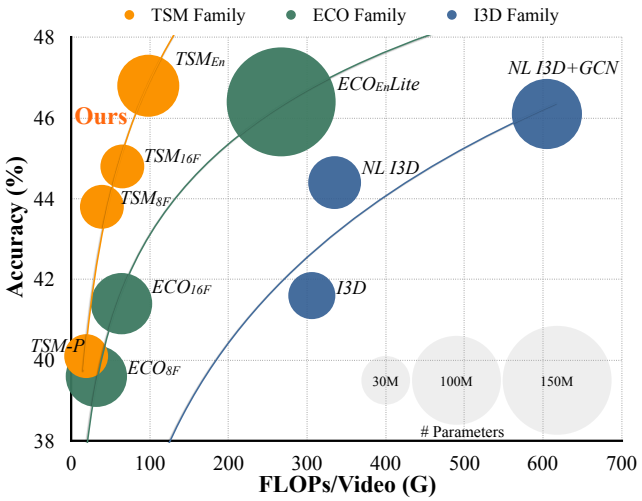


Figure 1. TSM is a *high performance* temporal modeling module. It achieves state-of-the-art performance on Something-Something-V1 [12] dataset which focus on temporal modeling. TSM is also a *high efficiency* module. It consumes $3\times$ less computation than the ECO family [53], $6\times$ less computation than the Non-local I3D family [43, 44] while having better performance. The model size is also $3\times$ smaller than the best ECO model. (GCN includes the cost of ResNet-50 RPN to generate region proposals.)

where N is the batch size, C is the number of channels, T is the temporal dimension, H and W are the spatial resolutions. 2D CNNs operate independently over the dimension T, so no temporal modeling happens. Our Temporal Shift module shifts the channels along the temporal dimension, both forward and backward. As shown in Figure 2, the information from neighboring frames are mingled with the current frame after shifting. Our intuition is: convolution operation consists of *shift* and *multiply-accumulate*. We *shift* in the time dimension by ± 1 and fold the *multiply-accumulate* from time dimension to channel dimension. The equivalence is temporal convolution with kernel size of 3. In terms of implementation, we only need to move the address pointer, not the data, thus efficiency is guaranteed.

We insert the Temporal Shift module to ResNet [14] to build our TSM video model (Figure 3). We follow [42, 53, 51] to divide the video into N segments and sample one frame from each segment. The sampled frames span the whole video, enabling long-term temporal relationship modeling. 2D CNN is then applied to each frame to extract spatial features. Within each 2D residual block, our Temporal Shift module shifts part of the channels along temporal dimension by ± 1 to fuse temporal information. The shifted activation is then processed by the following 2D convolution along the channel dimension. We find it beneficial to insert the Temporal Shift module into the residual branch rather than the outside so that it does not harm the spatial feature learning capability of the backbone 2D CNN. We also add an extra temporal max-pooling layer to reduce the temporal dimension, achieving even less computation compared to the pure 2D CNN.

We conducted extensive experiments to demonstrate the advantage of our TSM models. Firstly, our Temporal Shift module can improve the performance of 2D CNN on video recognition at no cost. On datasets where temporal relationships are important like Something-Something [12] and Jester [1], our method achieves 13% - 29% accuracy gain compared to pure 2D CNN; even on datasets where temporal relationships are less important, like UCF101 [34], HMDB51 [22], and Kinetics [21], our method achieves 2%-7% accuracy improvement. Secondly, our method is highly efficient. On Something-Something V1 dataset, TSM achieves the same accuracy compared to the previous state-of-the-art methods, while consuming only $5\times$ fewer FLOPs compared to non-local I3D [43, 44], $5.5\times$ fewer FLOPs and $6\times$ fewer parameters compared to ensemble ECO-Lite [53] (Figure 1). Thirdly, our TSM achieves high performance. At the time of submission, our method achieves the first place on both Something-Something V1 and V2 test leaderboards.

2. Related Work

2.1. Deep Video Recognition

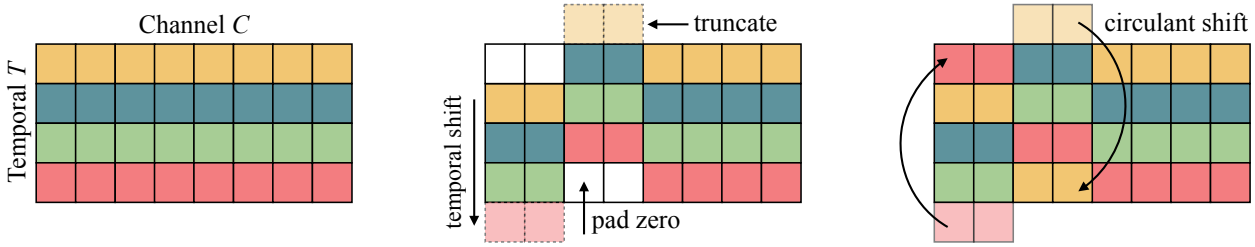
2D CNN. Using the 2D CNN is a straightforward way to conduct video recognition [20, 33, 42, 9, 6, 7, 2]. For example, Simonyan *et al*. [33] designed a two-stream CNN for RGB input (spatial stream) and optical flow [48] input (temporal stream) respectively. Temporal Segment Networks (TSN) [42] extracted averaged features from strided sampled frames. Such methods are more efficient compared to 3D counterparts. However, since the obtained features are fused using weighted average or simple mean pooling, the model cannot infer the temporal order or more complicated temporal relationships. For example, on Something-Something dataset where labels are related to the temporal modeling, TSN based method achieves top-1 accuracy less than 20%.

3D CNN. 3D convolutional neural networks can jointly learn spatio-temporal features. Tran *et al*. [39] proposed a 3D CNN based on VGG models, named C3D, to learn spatio-temporal features from a frame sequence. Carreira and Zisserman [3] proposed to inflate all the 2D convolution filters in an Inception V1 model [37] into 3D convolutions so that ImageNet pre-trained weights can be exploited for initialization. However, 3D CNNs are computationally heavy, making the deployment difficult. They also have more parameters than 2D counterparts, thus are more prone to over-fitting. On the other hand, our TSM has the same spatial-temporal modeling ability as 3D CNN while enjoying the same computation and parameters as the 2D CNNs.

Trade-offs. There have been attempts to trade off expressiveness and computation costs. Tran *et al*. [40] and Xie *et al*. [46] proposed to study mixed 2D and 3D networks, either first using 3D and later 2D (bottom-heavy) or first 2D and later 3D (top-heavy) architecture. ECO [53] also uses a similar top-heavy architecture to achieve a very efficient framework. Another way to save computation is to decompose the spatial and temporal dimension of the 3D convolution [40, 28, 36], which consists of a 2D spatial convolution and a 1D temporal convolutions. For mixed 2D-3D CNNs, they still need to remove low-level temporal modeling or high-level temporal modeling. Compared to decomposed convolutions, our method completely removes the cost of temporal modeling.

2.2. Temporal Modeling

Long-term temporal modeling is challenging. A direct way is to use 3D CNN based methods as discussed above. Wang *et al*. [43] proposed a spatial-temporal non-local module to capture long-range dependencies. Wang *et al*. [44] proposed to represent videos as space-time region graphs to capture both similarity relationships and spatial-temporal relationships. An alternative way to model the temporal relationships is to use 2D CNN + post-hoc fusion [11, 7, 51, 5].



(a) The original tensor without temporal fusion. Row_i only contains the feature maps from T_i.

(b) Tensor with temporal shift (zero padding). Row_i contains the feature maps from T_{i-1}, T_i and T_{i+1} (except boundary).

(c) Tensor with temporal shift (circulant padding). Row_i contains the feature maps from T_{i-1}, T_i and T_{i+1}.

Figure 2. Temporal shift module. For an activation tensor $A \in \mathbb{R}^{N\times T\times C\times H\times W}$, we display the T and C dimensions here. For a conventional 2D CNN, it operates on different frames independently (each frame is a row with the same color). While with our Temporal Shift module, adjacent three frames are mingled together (multiple colors per row). Temporal shift module has zero parameter overhead, zero computation overhead, yet effectively mixes the temporal information from adjacent frames.

Some works introduced LSTM [16] to aggregate the CNN features for video recognition [47, 5, 35, 8, 10]. Attention mechanism also proves to be effective for temporal modeling [31, 23, 26]. Zhou *et al*. [51] proposed Temporal Relation Network to learn and reason about temporal dependencies between video frames. The former category is computational heavy, while the latter cannot capture the useful low-level information that is lost during feature extraction. Our method offers an efficient solution at the cost of 2D CNNs, while enabling both low-level and high-level temporal modeling, just like 3D-CNN based methods.

2.3. Efficient Deep Models

The efficiency of 2D CNN has been extensively studied. Some works focused on designing an efficient model [18, 17, 30, 49]. Recently AutoML [54, 55, 25] has been introduced to find an efficient architecture automatically [38]. Another way is to compress or quantize an existing model for efficient deployment [13, 52, 15]. Address shift, which is a hardware-friendly primitive, has also been exploited for compact 2D CNN design on image recognition tasks [45, 50]. Nevertheless, building an efficient deep model for video understanding has received relatively less attention. In this paper, we propose an efficient module to enable any 2D CNN to efficiently conduct spatial-temporal feature learning for video recognition so that video understanding can also benefit from previous efforts on efficient 2D CNNs.

3. Approach

In this section, we will first describe the Temporal Shift Module (TSM) for temporal modeling, and then show how we use it to build the *TSM video model* for efficient long-term temporal video understanding.

3.1. Intuition

For video understanding framework, the shape of activation is usually $[N, C, T, H, W]$ (activation $A \in \mathbb{R}^{N\times C\times T\times H\times W}$), where N is the batch size, C is the number of channels, T is the temporal dimension, H and W are the spatial resolutions.

Let us first consider a normal convolution operation. For brevity, we used a 1-D convolution with the kernel size of 3 as an example. Suppose the weight of the convolution is $W = (w_1, w_2, w_3)$, and the input X is a 1-D vector with infinite length. The convolution operator $Y = \text{Conv}(W, X)$ can be written as: $Y_i = w_1 X_{i-1} + w_2 X_i + w_3 X_{i+1}$. We can decouple the operation of convolution into two steps: *shift* and *multiply-accumulate*: we shift the input X by $-1, 0, +1$ and multiply by w_1, w_2, w_3 respectively, which sum up to be Y. Formally, the *shift* operation is:

$$X_i^{-1} = X_{i-1}, \quad X_i^0 = X_i, \quad X_i^{+1} = x_{i+1} \tag{1}$$

and the *multiply-accumulate* operation is:

$$Y = w_1 X^{-1} + w_2 X^0 + w_3 X^{+1} \tag{2}$$

The first step *shift* can be conducted at no cost since it only needs an offset address pointer. While the second step is more computationally expensive, our Temporal Shift module *merges* the *multiply-accumulate* into the following 2D convolution, so it introduces no extra cost compared to 2D CNN based models.

3.2. Temporal Shift Module

Our proposed Temporal Shift module is described in Figure 2. For brevity, we only show the temporal T and channel C dimensions, while dropping batch size N, height H, and width W, since they are not related to our discussion. In Figure 2a, we describe a tensor with C channels and T frames. The features at different time stamps are denoted as different

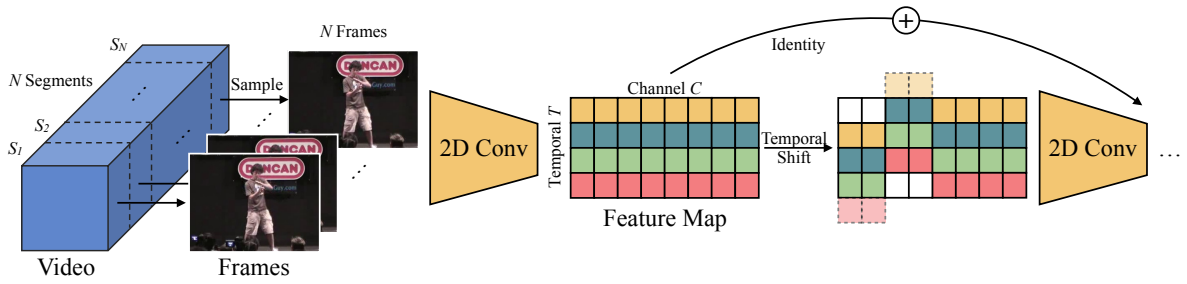
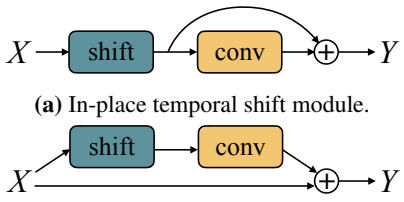


Figure 3. The architecture of TSM framework. The video is splitted into N segments of equal size. One frame is sampled from each segment. We use 2D convolution to extract the spatial feature from each frame. The Temporal Shift module is inserted afterwards to conduct temporal fusion at no cost.



(a) In-place temporal shift module.

(b) Residual temporal shift module.

Figure 4. Residual shift is better than in-place shift. In-place shift happens before a convolution layer (or a residual block). Residual shift fuses temporal information inside a residual branch.

colors in each row. Conventional 2D CNN operates among channels, that is to operate along each row separately without temporal fusion.

Along the temporal dimension, we shift one-quarter channels by -1, another quarter by $+1$, leaving the rest half un-shifted (Figure 2b). "A quarter" is a hyperparameter we will discuss in the ablation study (Section 3.3). Which quarter to shift doesn't matter since the weights in the next layer will adapt to it during training. As a result, we mingle the temporal information from frame $t-1$, t, and $t+1$ together, similar to a 1D convolution with the kernel size of 3, but at zero computation cost. The *multiply-accumulate* is pushed to the next layer. It borrows the 2D convolution's inter-channel fusion ability to perform the original inter-temporal fusion.

3.3. Variant of Temporal Shift

Here we describe the several variants of implementing our proposed Temporal Shift module.

In-place shift or residual shift. A straightforward way to apply our Temporal Shift module is to insert it before each convolutional layer or residual block, as illustrated in Figure 4a. We call such Temporal Shift module directly inserted before a convolution layer *in-place shift*.

One possible drawback of the *in-place shift* is the loss of spatial feature learning capability. After temporal shift, part of the information stored in the shifted channels is lost for the current frame. It could be a problem if we shift too many channels, *e.g.*, $1/2$ (ablation study in Section 4.5). To address such issue, we propose a variant of the shift module. Instead of inserting it in-place, we put the temporal shift module *inside* the residual branch in a residual layer, so that the information inside the shifted channels will still be available through identity mapping. We denote such version of shift as *residual shift*. Experiments show that residual based temporal shift achieves better performance compared to its in-place counterparts and is less sensitive to the proportion. Meanwhile, it does not harm learning spatial features while learning temporal features.

Zero padding or circulant padding. As illustrated in Figure 2, after temporal shifting, there will be empty channels at the boundary. We need to pad the empty boundary. The straightforward idea is to pad the empty space with zeros. The extra channels can be shifted out and truncated.

Another padding strategy is to use a circulant padding scheme, *i.e.*, the shifted-out channels are used to fill the corresponding empty spaces. However, the temporal order will be broken after circulant padding: the channel in the later frames will be shifted to the front of the early ones. Such padding strategy harms the temporal modeling performance. We verified this phenomenon in the ablation study (Section 4.5).

3.4. TSM Video Models

In this section, we describe how to leverage the proposed Temporal Shift module to build a video recognition model based on purely 2D CNNs without any extra cost.

Frame sampling. A raw video V usually consists of hundreds or thousands of frames with a high frame rate, which are redundant and too heavy to perform per frame inference. Thus video understanding models begin with frame sampling. Existing frame sampling strategy can be roughly divided into two categories: dense sampling [39, 3, 43, 46] and strided sampling [42, 51, 53]. To build an efficient model, we use strided sampling strategy which first divides the whole video into N sections of equal size $S_i, i = 1, 2, ..., N$, and samples one frame per section, same for training and testing. Another advantage of strided sampling is that it covers the whole video instead of a local subset; thus it is better for long-term

temporal modeling.

TSM Video Model. After frame sampling, 2D CNN baselines process each of the frames individually, and the output logits are averaged to give the final prediction. Our proposed TSM model is largely based on the corresponding 2D CNN baseline, with exactly the same parameters and computation cost. During most of the inference process like convolution layers, the frames are still running independently, just like the 2D CNNs. The difference is that for TSM models, temporal shift module is inserted for each layer/residual unit, which enables temporal information fusion at no cost. For each inserted temporal shift module, the temporal receptive field will be enlarged by 2, as if running a convolution with the kernel size of 3 along the temporal dimension. Therefore, our TSM model has a very large temporal receptive field to conduct highly complicated temporal modeling.

A unique advantage of our method is that we can easily convert any off-the-shelf 2D CNN model into a pseudo-3D model that can handle both spatial and temporal information, without adding additional cost. Thus the deployment of our framework is hardware friendly: we only need to support the operations in 2D CNNs, which are already well-optimized at both framework level (CuDNN [4], MKL-DNN) and hardware level (CPU/GPU/TPU/FPGA). The computation cost grows linearly with the number of frames. Previous methods [39, 3, 43, 44, 46] widely used temporal max pooling to reduce the temporal dimension during inference, which effectively reduces the computational cost of the network. We also used a similar technique here by inserting a max pooling layer between the first and the second residual layer. We use the kernel size of 3 and the stride of 2 for the temporal max pooling module. We denote the model without temporal pooling as **TSM**, and the model with temporal pooling as **TSM-P**.

4. Experiments

In this section, we show that TSM can significantly improve the performance of 2D CNN on video understanding at no cost. Then we demonstrate our TSM can achieve state-of-the-art performance on very temporal-related tasks, even better than 3D CNN based methods. We achieve the first place on both Something-Something V1 and V2 upon this paper's submission. Finally, our model is not only theoretically efficient but also achieves real speed up measured on GPU, with both lower latency and higher throughput.

4.1. Setups

Training. To evaluate the effectiveness of the proposed method, we conducted experiments on several action recognition datasets. Since many of the action recognition datasets are not large enough and are prone to over-fitting [42], we followed the common practice [43, 53, 3, 46] to first pre-trained the models on Kinetics [21] for 45 epochs with initial learning rate 0.01 and decayed by 0.1 every 15 epochs. We then fine-tuned the model to other target datasets like Something-Something [12], UCF101 [34], and HMDB51 [22]. The fine-tuning is conducted for 25 epochs with initial learning rate 0.001 and decayed by a factor of 0.1 every 10 epochs. We follow the practice in [42] to fix all the Batch Normalization [19] layers except for the first one.

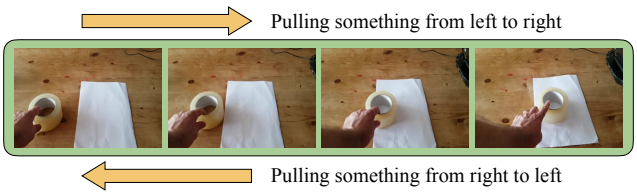


Figure 5. Something-Something dataset is temporal-related. Following different arrows of time gives different labels of the dataset.

We used 8 or 16 sampled frames per video for both training and testing. We applied similar data augmentation strategy as [42] for pre-processing: first resizing the raw images such that the shorter side is 256 and then employing a fixed corner+center cropping and scale-jittering. The images are then resized to 224×224 before being fed to the network. Unless otherwise specified, for all the training in our experiments, we used weight decay of 5e-4 and mini-batch SGD optimizer with momentum 0.9.

Testing. Many state-of-the-art methods use a different post-processing technique during the testing stage. For example, C3D [39] and I3D [3] sample multiple subsets of the video densely and give the averaged output as the prediction. TSN [42] and ARTNet [41] sample 25 frames from the image and apply 10 crops per frame leading to 250 processed frames per video. Since we target at making the inference efficient and fast, we only apply center crop using the same number of frames as training (*e.g.*, 8 or 16) to give the prediction directly unless otherwise specified.

Model. To have an apple-to-apple comparison with the state-of-the-art method [44], we used the same backbone (ResNet-50) on the same dataset (Something-Something-V1 [12]).This dataset focuses on temporal modeling. The difference is that [44] used 3D ResNet-50, while we used 2D ResNet-50 as the backbone to demonstrate efficiency.

We discussed several variants of our TSM model in Section 3.3. In our experiments, we used the best setting found by our ablation study (Section 4.5): residual temporal shift module with zero padding. We inserted such shift module into each residual unit of the backbone network. A quarter of the channels are shifted.

Datasets. Kinetics dataset [21] is a large-scale action recognition dataset with 400 classes. As pointed in [51, 46], datasets like Something-Something (V1&V2) [12], Charades [32], and Jester [1] are more focused on modeling the temporal relationships, while UCF101 [34], HMDB51 [22], and Kinetics [21] are less sensitive to temporal relationships.

Table 1. Our method consistently outperforms 2D counterparts on multiple datasets at zero extra cost. All the results are reported on the validation set using RGB modality and only 1 crop for efficiency.

Dataset	#Frame	Model	Acc1	Acc5	Δ Acc1
Kinetics	8	TSN	66.8	87.2	+3.8
		Ours	**70.6**	**89.5**	
	16	TSN	67.8	87.6	+4.7
		Ours	**72.5**	**90.7**	
UCF101	8	TSN	91.5	99.2	+2.5
		Ours	**94.0**	**99.2**	
	16	TSN	91.4	99.2	+3.1
		Ours	**94.5**	**99.5**	
HMDB51	8	TSN	63.2	88.2	+7.1
		Ours	**70.3**	**92.4**	
	16	TSN	63.6	88.5	+7.1
		Ours	**70.7**	**91.8**	
Something V1	8	TSN	19.7	46.6	+23.7
		Ours	**43.4**	**73.2**	
	16	TSN	19.9	47.3	+24.9
		Ours	**44.8**	**74.5**	
Something V2	8	TSN	27.8	57.6	+28.9
		Ours	**56.7**	**83.7**	
	16	TSN	30.0	60.5	+27.5
		Ours	**57.5**	**84.6**	
Jester	8	TSN	81.0	99.0	+13.4
		Ours	**94.4**	**99.7**	
	16	TSN	82.3	99.2	+13.0
		Ours	**95.3**	**99.8**	

Since our proposed method is a zero-cost approach to conduct temporal modeling, we mainly focus on the dataset with stronger temporal relationships like Something-Something (see Figure 5). Nevertheless, we also observed strong results on the other datasets and reported it.

4.2. Improving 2D CNN Baselines

We can seamlessly inject Temporal Shift module into a normal 2D CNN and improve its performance on video recognition. In this section, we demonstrate a 2D CNN baseline can significantly benefit from our Temporal Shift module with double-digits accuracy improvement. Here we aim to show the improvement over the baselines. We will demonstrate how to push it to the state-of-the-art in the next section. We used 8 and 16 sampled frames for training and testing, and used only center crop, which is a practical setup for high efficiency and fast inference application scenario.

We chose TSN [42] as the 2D CNN baseline. We used the same training protocol for TSN and our TSM. The only difference is whether our Temporal Shift module is used or not. The results on several action recognition datasets are listed in Table 1. The chart is split into two parts. The upper part contains datasets Kinetics [21], UCF101 [34], HMDB51 [22], where temporal relationships are less important, while our TSM still consistently outperforms the 2D TSN baseline at no extra cost. For the lower part, we present the results on Something-Something V1 and V2 [12] and Jester [1], which depend heavily on temporal relationships. TSN baseline cannot achieve a good accuracy, but once equipped with our Temporal Shift module, the performance improved by double digits. For example, on Something-Something-V1 dataset, our TSM achieves 24.9% better performance than the 2D baseline, even though the computation costs of the two methods are exactly the same.

4.3. Comparison with State-of-the-Arts

TSM not only significantly improves the 2D baseline but also outperforms state-of-the-art methods, which heavily rely on 3D convolutions. We compared the performance of our TSM model with state-of-the-art methods on both Something-Something V1&V2 because these two datasets focus on temporal modeling.

Something-Something-V1. Something-Something-V1 is a challenging dataset, as activity cannot be inferred merely from individual frames (*e.g.*, pushing something from *right to left*, pushing something from *left to right*, moving something *up*). An individual frame can not differ *left to right* and *right to left*, or *up* and *down*. We compared the results of the proposed TSM with current state-of-the-art methods on Something-Something-V1 in Table 2. We only applied the center crop during testing to ensure the efficiency.

In Table 2 we first present the 2D based methods, including TSN [42] and TRN [51]. TSN with different backbones fails to achieve decent performance (19.7% Top-1), due to the lack of temporal modeling. For TRN, although late temporal fusion is added after feature extraction, the result is still significantly lower than state-of-the-art methods', showing the importance of early temporal fusion.

The second section shows the state-of-the-art efficient video understanding framework ECO [53]. Compared to ECO, our method achieves better performance at a smaller FLOPs. For example, when using 16 frames as input, our TSM-P achieves 43.8% top-1 accuracy with 39G FLOPs, which is 2.4% more accurate than ECO and 1.6× less computation. The ensemble versions of ECO (ECO$_{En}$*Lite* and ECO$_{En}$*Lite*$_{\text{RGB+Flow}}$, using an ensemble of {16, 20, 24, 32} frames as input) did achieve competitive results, but the computation and parameters are too large for deployment, even surpassing I3D models. While our model is much more efficient: we only used {8, 16} frames model for ensemble (TSM$_{En}$), and the model achieves better performance using

Table 2. Comparison of TSM against other methods on Something-Something dataset. ⋆ *denotes the previous best result on the leaderboard.* * *includes parameters and FLOPs of the Region Proposal Network.*

Model	Backbone	#Frame	FLOPs	#Param.	Val Top-1	Val Top-5	Test Top-1
TSN [51]	BNInception	8	16G	10.7M	19.5	-	-
TSN (our impl.)	ResNet-50	8	33G	24.3M	19.7	46.6	-
TRN-Multiscale [51]	BNInception	8	16G	18.3M	34.4	-	33.6
Two-stream $\text{TRN}_{\text{RGB+Flow}}$ [51]	BNInception	8+8	-	36.6M	42.0	-	40.7
ECO [53]	BNInception+ 3D ResNet-18	8	32G	47.5M	39.6	-	-
ECO [53]		16	64G	47.5M	41.4	-	-
$\text{ECO}_{En}\textit{Lite}$ [53]		92	267G	150M	46.4	-	42.3
$\text{ECO}_{En}\textit{Lite}_{\text{RGB+Flow}}$ [53]		92+92	-	300M	49.5	-	43.9
I3D [44]	3D ResNet-50	64	306G	28.0M	41.6	72.2	-
Non-local I3D [44]	3D ResNet-50	64	335G	35.3M	44.4	76.0	-
⋆Non-local I3D + GCN [44]	3D ResNet-50+GCN	64	605G*	62.2M*	46.1	76.8	45.0
TSM-P	ResNet-50	8	19G	24.3M	40.1	69.2	-
TSM-P	ResNet-50	16	39G	24.3M	43.8	73.4	-
TSM	ResNet-50	8	33G	24.3M	43.4	73.2	-
TSM	ResNet-50	16	65G	24.3M	44.8	74.5	-
TSM_{En}	ResNet-50	24	98G	48.6M	46.8	76.1	-
$\text{TSM}_{\text{RGB+Flow}}$	ResNet-50	16+8	-	48.6M	**49.6**	**79.0**	**46.1**

Table 3. Results on Something-Something-V2. Our TSM achieves comparable performance to previous state-of-the-art results using only RGB input. Our two-stream method outperforms previous state-of-the-art on the leaderboard by 3.6% without bells and whistle. ⋆ *denotes the previous best result on the leaderboard.*

Method	Val		Test	
	Top-1	Top-5	Top-1	Top-5
TSN (our impl.)	30.0	60.5	-	-
MultiScale TRN [51]	48.8	77.6	50.9	79.3
2-Stream TRN [51]	55.5	83.1	56.2	83.2
⋆Two-stream Dual Attention Network	-	-	60.1	86.1
TSM_{8F}	58.2	84.8	-	-
TSM_{16F}	58.7	84.8	59.9	85.9
$\text{TSM}_{\text{RGB+Flow}}$	**63.5**	**88.6**	**63.7**	**89.5**

2.7× less computation and 3.1× fewer parameters.

The third section contains methods that achieve previous best performance on the leaderboard: Non-local I3D + GCN [44] (marked with ⋆ in the table). The GCN needs a Region Proposal Network [29] trained on MSCOCO object detection dataset [24] to generate the bounding boxes, which is unfair to compare since external data (MSCOCO) and extra training cost is introduced. Thus we compared TSM to its CNN part: Non-local I3D. Our TSM (16f) achieves 0.4% better accuracy with 5× fewer FLOPs on the validation set compared to the Non-local I3D network. Note that techniques like Non-local module [43] are orthogonal to our work, which could also be added to our framework to boost the performance further.

Generalize to Other Modalities. We also show that our proposed method can generalize to other modalities like optical flow. To extract the optical flow information between frames, we followed [42] to use the TVL1 optical flow algorithm [48] implemented in OpenCV with CUDA. We conducted two-stream experiments on both Something-Something V1 and V2 datasets, and it consistently improves over the RGB performance: introducing optical flow branch brings 4.8% top-1 improvement on both V1 and V2 evaluation sets. Our two-stream model achieves the *current best result* on both Something-Something V1 and V2 leaderboard at the time of submission.

Something-Something-V2. We also show the result on Something-Something-V2 dataset, which is a newer release to its previous version, containing more videos for training. The results compared to other state-of-the-art methods are shown in Table 3. Here we aim to show the high performance of our TSM model. Therefore, we followed [42] that used 5-crop test protocol (center and four corners) to evaluate the model, which leads to around 1% performance gain. On Something-Something-V2 dataset, we achieved the current

Table 4. TSM consistently outperforms ECO on Something-Something dataset, while consumes 3.7× less GPU memory, 1.5× better GPU inference latency (measured with batch size = 1), 1.7× better GPU inference throughput (measured with batch size = 16). V/s means videos per second, higher the better. Measured on NVIDIA Tesla P100 GPU.

Model	FLOPs	Memory	Latency	Throughput	Sth Acc.
I3D [44]	306G	540MB	165.3ms	6.1V/s	41.6%
ECO_{16F} [53]	64G	735MB	30.6ms	45.6V/s	41.4%
TSM_{8F}	**33G**	**199MB**	**20.7ms**	**77.1V/s**	43.4%
$TSM\text{-}P_{16F}$	39G	301MB	25.7ms	54.8V/s	43.8%
TSM_{16F}	65G	301MB	32.5ms	39.5V/s	**44.8%**

best result on the leaderboard at the time of submission without bells and whistle, surpassing previous state-of-the-art result (denoted with ⋆) by 3.6%. Note that the RGB input result of our model is only 0.2% worse than previous state-of-the-art result using two-stream input, showing that our TSM is strong at handling temporal modeling.

Cost *vs*. Accuracy. Our TSM model achieves very competitive performance while enjoying high efficiency and low computation cost for fast inference. We show the FLOPs for each model in Table 2. Although GCN itself is light, the method used a ResNet-50 based Region Proposal Network [29] to extract bounding boxes, whose cost is also considered in the chart. Note that in actual deployment we need to consider the computation cost of optical flow extraction, which is usually much larger than the video recognition model itself. Therefore, we do not report the FLOPs of two-stream based methods.

We show the accuracy, FLOPs, and number of parameters trade-off in Figure 1. The accuracy is tested on the validation set of Something-Something-V1 dataset, and the number of parameters is indicated by the area of the circles. We can see that our TSM based methods have a better Pareto curve than both previous state-of-the-art efficient models (ECO based models) and high-performance models (non-local I3D based models). Our TSM models are both efficient and accurate. It can achieve state-of-the-art accuracy at high efficiency: it achieves better performance while consuming 3× less computation than the ECO family and 6× less computation than the Non-local I3D family. The model size is also 3× smaller than ensemble ECO. Considering that ECO is already an efficiency-oriented design, our method enjoys highly competitive efficiency.

4.4. Latency and Throughput Speedup

The measured inference latency and throughput are important for the large-scale deployment of video understanding algorithms. Our method has not only low FLOPs but also low latency and high throughput. We performed measurement on a single NVIDIA Tesla P100 GPU. We excluded the data loading time for fair comparison. To measure the latency, we used a batch size of 1; to measure the throughput, we used a batch size of 16. We first conducted warm-up inference for 100 runs and measured the averaged speed for 200 runs. The memory consumption was measured with PyTorch [27]. Compared with the popular I3D algorithm, our method achieved 8× lower latency (20.7ms *vs*. 165.3ms), 12× higher throughput (77 videos per second *vs*. 6 videos per second), 2.7× smaller memory consumption (199MB *vs*. 540MB), at 1.8% higher accuracy (Table 4). We also compared our method to the state-of-the-art efficient model ECO [53]: Our TSM-P model has 1.5× lower latency (20.7ms *vs*. 30.6ms), 1.7× higher throughput, less than half the memory consumption, and achieves 2% better accuracy. ECO has an expensive two-branch (2D+3D) architecture, while our method only needs the in-expensive 2D backbone. Furthermore, the above results are based on a naive split-pad-concatenate implementation. We expect even higher speedup if we can implement a customized kernel for temporal shift.

Table 5. Residual shift is better than in-place shift. In-place shift will harm the model capacity on spatial feature learning. Residual shift can balance between the spatial and temporal feature extraction. The performance is evaluated on the validation set of Kinetics.

Architecture	Proportion	Acc-1	Acc-5
TSN baseline	-	66.8	87.2
In-place shift	1/4	69.7	89.3
	1/2	67.8	88.3
Residual shift	1/8	70.4	**89.6**
	1/4	**70.6**	89.5
	1/2	69.9	89.7

Table 6. Zero padding is better than circulant padding.

Padding Type	Kinetics Acc.	Something Acc.
TSN baseline	66.8	19.7
Zero	**70.6**	**43.4**
Circulant	69.5	40.8

4.5. Ablation Study

We conducted ablation study to find the best implementation of Temporal Shift module as discussed in Section 3.3.

In-place *vs*. Residual. We conducted experiments on shift *place* and the *proportion* of channels to shift on Kinetics [21] dataset using 8 frames. The result is shown in Table 5. The first finding is that residual shift has better performance than in-place shift. This is because some of the information is lost in in-place shifted channels. While residual based tempo-

ral shift can still access the information of shifted channels through identity mapping. Another finding is that the performance is related to the proportion of shifted channels: if the proportion is too small, the ability of temporal reasoning may not be enough to handle complicated temporal relationships; if too large, the spatial feature learning ability may be hurt. For residual shift, we found that the performance reached the peak when $1/4$ of the channels were shifted. Therefore, we use this setting throughout our experiments.

Padding. We compared the zero padding and circulant padding on Kinetics and Something-Something-V1 dataset (Table 6). Circulant padding hurts the performance due to temporal disorder: an old frame will be ahead of an early frame. Therefore we choose zero padding for our framework.

5. Conclusion

In this paper, we proposed a Temporal Shift module that can be inserted into 2D CNN backbone to enable joint spatial-temporal modeling at no additional cost. The module shifts part of the channels along temporal dimension bi-directionally to exchange information with neighboring frames. Our framework is both efficient and accurate for video recognition. On Something-Something-V1 dataset we achieved better results than I3D family and ECO family using $6\times$ and $2.7\times$ fewer FLOPs respectively. Upon this paper's submission, we achieved the state-of-the-art results on both Something-Something V1 and V2 leaderboards.

Acknowledgments We thank MIT Quest for Intelligence, MIT-IBM Watson AI Lab, SenseTime, Google Cloud and Snap for supporting this research. We thank Bowen Pan for data processing.

References

[1] The 20bn-jester dataset v1. https://20bn.com/datasets/jester. 2, 5, 6

[2] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3034–3042, 2016. 2

[3] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 4724–4733. IEEE, 2017. 1, 2, 4, 5

[4] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014. 5

[5] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015. 1, 2, 3

[6] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in neural information processing systems*, pages 3468–3476, 2016. 2

[7] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1933–1941, 2016. 1, 2

[8] C. Gan, C. Sun, L. Duan, and B. Gong. Webly-supervised video recognition by mutually voting for relevant web images and web video frames. In *European Conference on Computer Vision*, pages 849–866. Springer, 2016. 3

[9] C. Gan, N. Wang, Y. Yang, D.-Y. Yeung, and A. G. Hauptmann. Devnet: A deep event network for multimedia event detection and evidence recounting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2568–2577, 2015. 2

[10] C. Gan, T. Yao, K. Yang, Y. Yang, and T. Mei. You lead, we exceed: Labor-free video concept learning by jointly exploiting web videos and images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 923–932, 2016. 3

[11] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, volume 2, page 3, 2017. 1, 2

[12] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al. The something something video database for learning and evaluating visual common sense. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 1, page 3, 2017. 1, 2, 5, 6

[13] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 3

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[15] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018. 3

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 3

[18] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 3

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

[20] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 1, 2

[21] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 2, 5, 6, 8

[22] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2556–2563. IEEE, 2011. 2, 5, 6

[23] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. Snoek. Videolstm convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 166:41–50, 2018. 3

[24] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 7

[25] C. Liu, B. Zoph, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. *arXiv preprint arXiv:1712.00559*, 2017. 3

[26] X. Long, C. Gan, G. de Melo, J. Wu, X. Liu, and S. Wen. Attention clusters: Purely attention based local feature integration for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7834–7843, 2018. 3

[27] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 8

[28] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5534–5542. IEEE, 2017. 2

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 6, 8

[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 3

[31] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*, 2015. 3

[32] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016. 5

[33] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. 1, 2

[34] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 2, 5, 6

[35] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015. 3

[36] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4597–4605, 2015. 2

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 2

[38] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. *arXiv preprint arXiv:1807.11626*, 2018. 3

[39] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. 1, 2, 4, 5

[40] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018. 1, 2

[41] L. Wang, W. Li, W. Li, and L. Van Gool. Appearance-and-relation networks for video classification. *arXiv preprint arXiv:1711.09125*, 2017. 5

[42] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016. 1, 2, 4, 5, 6, 7

[43] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. *arXiv preprint arXiv:1711.07971*, 10, 2017. 1, 2, 4, 5, 7

[44] X. Wang and A. Gupta. Videos as space-time region graphs. *arXiv preprint arXiv:1806.01810*, 2018. 1, 2, 5, 6, 7, 8

[45] B. Wu, A. Wan, X. Yue, P. Jin, S. Zhao, N. Golmant, A. Gholaminejad, J. Gonzalez, and K. Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. *arXiv preprint arXiv:1711.08141*, 2017. 3

[46] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018. 1, 2, 4, 5

[47] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015. 3

[48] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint Pattern Recognition Symposium*, pages 214–223. Springer, 2007. 2, 7

[49] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CoRR*, abs/1707.01083, 2017. 3

[50] H. Zhong, X. Liu, Y. He, Y. Ma, and K. Kitani. Shift-based primitives for efficient convolutional neural networks. *arXiv preprint arXiv:1809.08458*, 2018. 3

[51] B. Zhou, A. Andonian, and A. Torralba. Temporal relational reasoning in videos. *arXiv preprint arXiv:1711.08496*, 2017. 1, 2, 3, 4, 5, 6, 7

[52] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016. 3

[53] M. Zolfaghari, K. Singh, and T. Brox. Eco: Efficient convolutional network for online video understanding. *arXiv preprint arXiv:1804.09066*, 2018. 1, 2, 4, 5, 6, 7, 8

[54] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 3

[55] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2(6), 2017. 3