TADA! TEMPORALLY-ADAPTIVE CONVOLUTIONS FOR VIDEO UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Spatial convolutions[1] are widely used in numerous deep video models. It fundamentally assumes spatio-temporal invariance, *i.e.*, using shared weights for every location in different frames. This work presents **Temporally-Adaptive Convolutions (TAdaConv)** for video understanding, which shows that adaptive weight calibration along the temporal dimension is an efficient way to facilitate modelling complex temporal dynamics in videos. Specifically, TAdaConv empowers the spatial convolutions with temporal modelling abilities by calibrating the convolution weights for each frame according to its local and global temporal context. Compared to previous temporal modelling operations, TAdaConv is more efficient as it operates over the convolution kernels instead of the features, whose dimension is an order of magnitude smaller than the spatial resolutions. Further, the kernel calibration also brings an increased model capacity. We construct TAda2D networks by replacing the spatial convolutions in ResNet with TAdaConv, which leads to on par or better performance compared to state-of-the-art approaches on multiple video action recognition and localization benchmarks. We also demonstrate that as a readily plug-in operation with negligible computation overhead, TAdaConv can effectively improve many existing video models with a convincing margin.

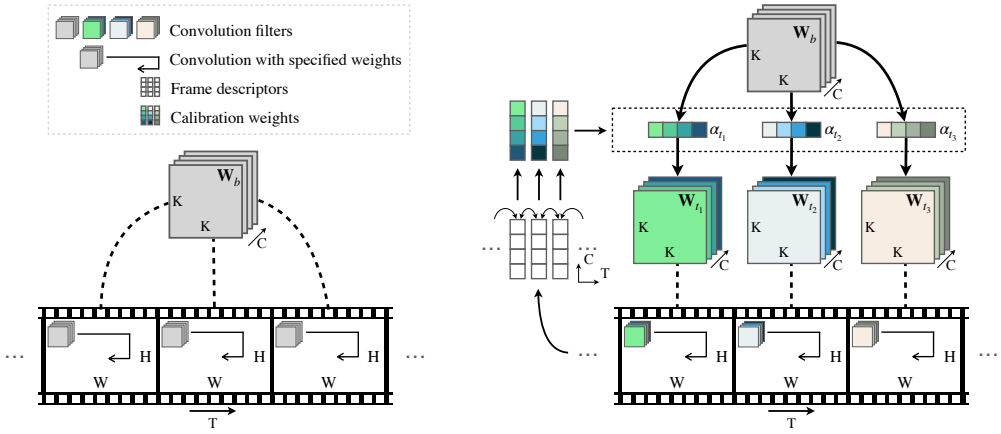
1 INTRODUCTION

Convolutions are an indispensable operation in modern deep vision models (He et al., 2016; Szegedy et al., 2015; Krizhevsky et al., 2012), whose different variants have driven the state-of-the-art performances of convolutional neural networks (CNNs) in many visual tasks (Xie et al., 2017; Chen et al., 2017; Dai et al., 2017; Zhou et al., 2019) and application scenarios (Howard et al., 2017; Yang et al., 2019). In the video paradigm, compared to its three-dimensional counterpart (Tran et al., 2015), 2D convolutions are actually widely preferred owing to its efficiency (Wang et al., 2016; Jiang et al., 2019; Lin et al., 2019a; Li et al., 2020b; Liu et al., 2021; Feichtenhofer et al., 2019).

One essential property of the spatial convolution (*i.e.*, 2D convolution) operation is its translation invariance (Ruderman & Bialek, 1994; Olshausen & Field, 1996; Bartunov et al., 2018), resulted from its local connectivity and shared weights. Because it can be simply generalized to the temporal dimension (Simoncelli & Olshausen, 2001), it is conventional for modern deep video models to share the 2D convolution weights at different time steps (Karpathy et al., 2014; Wang et al., 2016).

However, for a genuine understanding of videos, semantic variations along the temporal dimension has to be considered, which is not achievable for the spatial convolutions in its current form. To capture this diverse motion pattern, additional operations are required to be coupled with the spatial convolutions, such as the difference operator (Wang et al., 2021), the temporal convolutions (Tran et al., 2018) or relation-based operations (Wang et al., 2018a). It can be observed that these operators share the same underlying process: different weights are assigned to different frames in a local or a global perspective, before information from different time steps are aggregated. Essentially, these weights assigned to different time steps can be mathematically viewed as the calibration weights for the spatial convolution kernels. In light of this, we aim to achieve temporal modeling by directly integrating the dynamic weight calibration into the spatial convolutions.

[1] In this work, we use spatial convolutions and 2D convolutions interchangeably.



(a) Spatial convolution in videos

(b) TAdaConv in videos

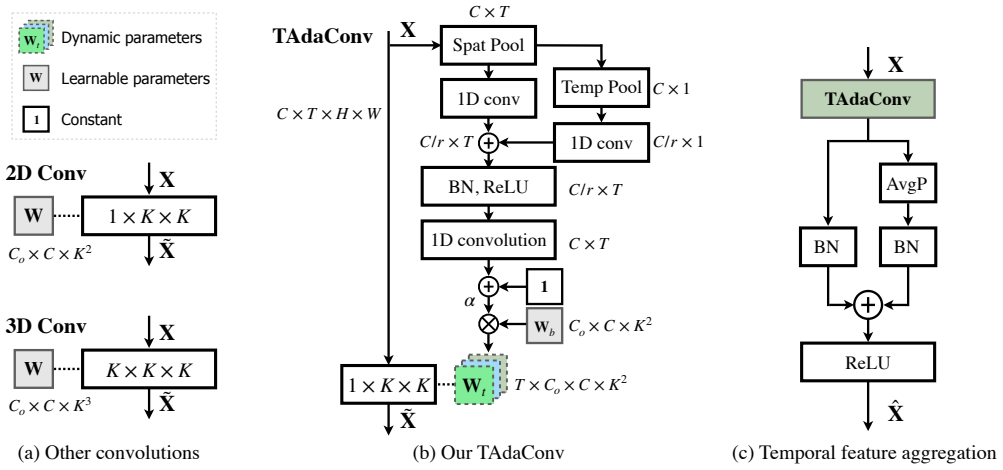
Figure 1: **Comparisons between TAdaConv and the spatial convolutions in video models**. (a) Conventional spatial convolutions in videos share the convolution weights between different frames. (b) Our proposed TAdaConv adaptively calibrates the kernel weights for each frame from a base weight with its adjacent frames and global context.

In this work, we present the Temporally-Adaptive Convolution (TAdaConv) for video understanding, where the convolution kernel weights are no longer fixed across different frames. Specifically, the convolution kernel for the t-th frame $\mathbf{W}_t$ is factorized to the multiplication of the base weight and a calibration weight: $\mathbf{W}_t = \boldsymbol{\alpha}_t \cdot \mathbf{W}_b$, where the calibration weight $\boldsymbol{\alpha}_t$ is adaptively generated from the input data for all channels in the base weight $\mathbf{W}_b$. For each frame, we generate the calibration weight based on the frame descriptors of its adjacent time steps as well as the global descriptor, which effectively encodes the local and global temporal dynamics in videos. The main advantages of this factorization are three-fold: **(i)** TAdaConv can be easily plugged into any existing models to enhance temporal modelling, and their pre-trained weights can still be exploited; **(ii)** the model capacity can be highly increased with the help of the content-adaptive calibration weight; **(iii)** in comparison with temporal convolutions that often operate on the learned 2D feature maps, TAdaConv is more efficient by directly operating on the convolution kernels.

TAdaConv is proposed as a direct replacement for the spatial convolutions. It can either serve as a stand-alone temporal modelling module building upon 2D networks, or be coupled with existing video models that use spatial convolutions to further enhance the model capacity as well as the ability to learn temporal dynamics. Constructed from ResNet (He et al., 2016) (*i.e.,* R2D), our TAda2D achieves on par or better performance than the state-of-the-arts. Used as an enhancement of existing models, notable improvements are observed on multiple video datasets. The strong performance and the consistent improvements demonstrate that the temporally adaptive weight calibration can be an important aspect to consider when modelling the complex temporal dynamics in videos.

2 Related Work

ConvNets for temporal modelling. A fundamental difference between videos and images lies in the temporal dimension, which makes temporal modeling an important research area for understanding videos. Recent deep CNNs for video understanding can be divided into two types. One expands the 2D convolutions in the temporal dimension, modeling spatio-temporal information jointly in one operation (Carreira & Zisserman, 2017; Tran et al., 2015; Feichtenhofer, 2020; Tran et al., 2019). The other builds upon 2D networks, and perform temporal modeling by additional operations such as temporal shift (Lin et al., 2019a), temporal difference (Wang et al., 2021; Jiang et al., 2019), temporal convolution (Jiang et al., 2019; Tran et al., 2018; Qiu et al., 2017; Liu et al., 2021) and correlation operation (Wang et al., 2020), *etc*. For the latter variant, which is the focus of this paper, most approaches employ 2D convolutions that shares weights among all the frames for modeling spatial semantics, and design additional operations for temporal modelling. This work directly empowers



(a) Other convolutions (b) Our TAdaConv (c) Temporal feature aggregation

Figure 2: **An instantiation of TAdaConv and the temporal feature aggregation used in TAda2D.** (a) Common convolutions used in video models. (b) Our TAdaConv using non-linear weight calibrations with global temporal context. (c) The temporal feature aggregation scheme used in TAda2D.

the spatial convolutions with temporal modelling abilities, which can be further coupled with other temporal modelling operations to achieve stronger video recognition performances.

Dynamic networks. Dynamic networks refer to the networks with content-adaptive weights or modules. It can have multiple forms, such as dynamic filters (Jia et al., 2016; Hu et al., 2018; Li et al., 2019), dynamic convolutions (Yang et al., 2019; Chen et al., 2020a; Li et al., 2021), dynamic activations (Li et al., 2020d; Chen et al., 2020b), and dynamic routing (Wang et al., 2018b; Li et al., 2020c), *etc*. Dynamic networks have demonstrated their exceeding network capacity and thus performance compared to the static ones. In the video paradigm, dynamic kernel also shows strong abilities to model temporal information (Liu et al., 2021), while the efficiency of the video models is improved by adaptive fusion along the temporal dimension (Meng et al., 2021). However, in most previous dynamic variants for convolutions, the convolution operation is employed in its original form, *i.e.,* shared weights in all input locations. Recently, some convolutions with location-sensitive weights have been proposed to model complex spatial semantics (Elsayed et al., 2020; Chen et al., 2021), which has shown the benefit of relaxing the spatial invariance. In comparison with these dynamic networks, the proposed TAdaConv has two key differences: (i) the convolution weight in TAdaConv is obtained by the multiplication of a base weight and a calibration weight; (ii) the convolution is performed with temporally-sensitive weights, relaxing the temporal invariance.

3 TAdaConv: Temporally-adaptive Convolutions

This work mainly considers the 2D convolutions in deep video models, which conventionally share weights for all the frames in a video. Inspired by the underlying calibration performed by temporal convolutions (Sec. 3.1), the proposed TAdaConv dynamically calibrates the convolution weights for each frame according to its adjacent frames and the global context (Sec. 3.2). The difference between TAdaConv and the spatial convolutions is visualized in Fig. 1.

3.1 Revisiting temporal convolutions

We first revisit the temporal convolutions to showcase its underlying process for temporal modelling. For simplicity, depth-wise temporal convolution is considered, which is also more widely applied because of its efficiency (Liu et al., 2021; Jiang et al., 2019; Li et al., 2020b). Specifically, for a $3\times1\times1$ temporal convolution filter parameterized by $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \boldsymbol{\beta}_3]$ and directly placed (ignoring normalizations and activations) after the spatial convolution parameterized by $\mathbf{W}$, the output feature $\tilde{\mathbf{x}}_t$ of the t-th frame can be obtained by:

$$\tilde{\mathbf{x}}_t = \boldsymbol{\beta}_1 \cdot (\mathbf{W} * \mathbf{x}_{t-1}) + \boldsymbol{\beta}_2 \cdot (\mathbf{W} * \mathbf{x}_t) + \boldsymbol{\beta}_3 \cdot (\mathbf{W} * \mathbf{x}_{t+1}) \,, \tag{1}$$

where the operator $\cdot$ indicates the element-wise multiplication and $*$ denotes the convolution over the spatial dimension. It can be rewritten as follows:

$$\tilde{\mathbf{x}}_t = \mathbf{W}_{t-1} * \mathbf{x}_{t-1} + \mathbf{W}_t * \mathbf{x}_t + \mathbf{W}_{t+1} * \mathbf{x}_{t+1} \ , \tag{2}$$

where $\mathbf{W}_{t-1} = \boldsymbol{\beta}_1 \cdot \mathbf{W}, \mathbf{W}_t = \boldsymbol{\beta}_2 \cdot \mathbf{W}$ and $\mathbf{W}_{t+1} = \boldsymbol{\beta}_3 \cdot \mathbf{W}$. Hence, the underlying process of temporal convolution can be divided into two steps: (i) assigning a different weight $\boldsymbol{\beta}_i$ for each frame, where $\boldsymbol{\beta}$ can be thought of as a calibration factor for the spatial convolution weights, and (ii) aggregating features from adjacent time steps.

3.2 Formulation of TAdaConv

Inspired by the underlying calibration performed by temporal convolutions, we set out to achieve temporal modelling by directly making the convolution weights adaptively calibrated along the temporal dimension. This means the convolution weights in a TAdaConv layer are varied on a frame-by-frame basis. Specifically, the weights in TAdaConv for the t-th frame $\mathbf{W}_t$ is factorized into the multiplication of a base weight $\mathbf{W}_b$ that is shared for all the frames, and a calibration weight $\boldsymbol{\alpha}_t$ that is temporally sensitive, *i.e.,* distinct for different frames. The output feature can thus be obtained by:

$$\tilde{\mathbf{x}}_t = \mathbf{W}_t * \mathbf{x}_t = (\boldsymbol{\alpha}_t \cdot \mathbf{W}_b) * \mathbf{x}_t \ , \tag{3}$$

where $\mathbf{W}_b \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times k^2}$, with C_{out}, C_{in} and k being the output dimension, input dimension and the kernel size for the spatial convolutions.

Calibration weight generation. To allow for the TAdaConv to model temporal dynamics, the calibration weight $\boldsymbol{\alpha}_t$ for the t-th frame is designed as a function of its local and global temporal context. It is crucial for the calibration weight $\boldsymbol{\alpha}$ to be generated based on the input, *i.e.,* $\boldsymbol{\alpha} = \mathcal{G}(\mathbf{x})$, because otherwise the TAdaConv layer would degenerate to a set of unrelated common spatial convolutions applied on different frames. An instantiation of the weight generation function $\mathcal{G}$ is visualized in Fig. 2(b). In our design, we aim for efficiency and the ability to capture inter-frame temporal dynamics with a sufficient field of view. For efficiency, we operate on the frame description vectors $\mathbf{v} \in \mathbb{R}^{T \times C}$ obtained by spatial global average pooling operation GAP_s on each frame. For temporal modelling, we first apply stacked two-layer 1D convolutions $\mathcal{F}$ with a dimension reduction ratio of r on the local temporal context $\mathbf{x}_t^{adj} = \{\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}\}$:

$$\mathcal{F}(\mathbf{x}_t^{adj}) = \text{Conv1D}^{C/r \to C}(\delta(\text{BN}(\text{Conv1D}^{C \to C/r}(\mathbf{v}_t^{adj})))) \ . \tag{4}$$

where $\mathbf{v}_t^{adj} = \{\text{GAP}_s(\mathbf{x}_{t-1}), \text{GAP}_s(\mathbf{x}_t), \text{GAP}_s(\mathbf{x}_{t+1})\}$. δ and BN respectively denote the ReLU activation (Nair & Hinton, 2010) and batch normalizations (Ioffe & Szegedy, 2015).

In order for a larger inter-frame field of view in complement to the local 1D convolution, we further incorporate global temporal information by adding a global descriptor $\mathbf{g}$ to the weight generation process $\mathcal{F}$ through a linear mapping function FC:

$$\mathcal{F}(\mathbf{x}_t^{adj}, \mathbf{g}) = \text{Conv1D}^{C/r \to C}(\delta(\text{BN}(\text{Conv1D}^{C \to C/r}(\mathbf{v}_t^{adj}) + \text{FC}^{C \to C/r}(\mathbf{g}))) \ , \tag{5}$$

where $\mathbf{g}$ is obtained through a global average pooling on the input feature, *i.e.,* $\mathbf{g} = \text{GAP}_{st}(\mathbf{x})$.

Initialization. The TAdaConv is designed to be readily inserted into existing models by simply replacing the 2D convolutions. In order to take advantage of the pre-trained weights, TAdaConv at its initial state is designed to behave exactly as the original convolution. Therefore, we initialize the weight of the last convolution in $\mathcal{F}$ to be zero, and adding a constant vector $\mathbf{1}$ to the formulation:

$$\boldsymbol{\alpha}_t = \mathcal{G}(\mathbf{x}) = \mathbf{1} + \mathcal{F}(\mathbf{x}_t^{adj}, \mathbf{g}) \ . \tag{6}$$

In this way, at initial state, $\mathbf{W}_t = \mathbf{1} \cdot \mathbf{W}_b = \mathbf{W}_b$, where we load $\mathbf{W}_b$ with the pre-trained weights.

Calibration dimension. The base weight can be calibrated in different dimensions. We instantiate the calibration on the C_{in} dimension ($\boldsymbol{\alpha}_t \in \mathbb{R}^{1 \times C_{in} \times 1}$), as the weight generation function over the global descriptors of the input features yields a more precise estimation for the relation of the input channels than the output channels or spatial structures. Empirical analysis is shown in Table 4.

Table 1 compares the TAdaConv with some of the other temporal modelling operations in parameters and FLOPs. Compared to (2+1)D convolution, most of our additional computation overhead on top of the spatial convolution is an order of magnitude less than the temporal convolution[2].

[2]For $C_o = C_i = 64, k = 3, T = 8, H = W = 56$ and $r = 4$, the computation cost for (2+1)D convolution and ours are respectively 1.2331 and 0.9268 GFLOPs. The computation overhead of TAdaConv over the spatial convolution (0.9248 GFlops) is only 0.002 GFLOPs.

Table 1: Comparison of different operations for spatial and temporal modeling (Lin et al., 2019a; Tran et al., 2018; Wang et al., 2020; Hara et al., 2018). 'T.' refers to the temporal modeling ability.

T.	Operation	Parameters	FLOPs
✗	Spat. conv	$C_o \times C_i \times k^2$	$C_o \times C_i \times k^2 \times THW$
✓	Temp. conv	$C_o \times C_i \times k$	$C_o \times C_i \times k \times THW$
✓	Temp. shift	$C_o \times C_i \times k^2$	$C_o \times C_i \times k^2 \times THW$
✓	(2+1)D conv	$C_o \times C_i \times (k^2 + k)$	$C_o \times C_i \times (k^2 + k) \times THW$
✓	3D conv	$C_o \times C_i \times k^3$	$C_o \times C_i \times k^3 \times THW$
✓	Correlation	$C_i \times T \times k^2$	$C_i \times k^2 \times THW$
✓	TAdaConv	$C_o \times C_i \times k^2 + 2 \times C_i \times C_i/r \times k$	$C_o \times C_i \times k^2 \times THW + C_i \times THW$ $+2 \times C_i \times C_i/r \times k \times T + C_o \times C_i \times k^2 \times T$

4 TAda2D: Temporally Adaptive 2D Networks

We construct TAda2D networks by replacing the 2D convolutions in ResNet (R2D, see Appendix B) with the proposed TAdaConv. Additionally, based on simple average pooling, we propose a temporal feature aggregation module for the 2D networks, corresponding to the second essential step for the temporal convolutions. As illustrated in Fig. 2(c), the aggregation module is placed after TAdaConv. Formally, given the output of TAdaConv $\tilde{\mathbf{x}}$, the aggregated feature can be obtained as follows:

$$\mathbf{x}_{aggr} = \delta(\mathrm{BN}_1(\tilde{\mathbf{x}}) + \mathrm{BN}_2(\mathrm{TempAvgPool}_k(\tilde{\mathbf{x}}))) \ , \tag{7}$$

where $\mathrm{TempAvgPool}_k$ denotes strided temporal average pooling with kernel size of k. We use different batch normalization parameters for the features extracted by TAdaConv $\tilde{\mathbf{x}}$ and aggregated by strided average pooling $\mathrm{TempAvgPool}_k(\tilde{\mathbf{x}})$, as their distributions are essentially different. During initialization, we load pre-trained weights to BN_1, and initialize the parameters of BN_2 as zero. Coupled with the initialization of TAdaConv, the initial state of the TAda2D is exactly the same as the Temporal Segment Networks (Wang et al., 2016), while the calibration and the aggregation notably increases the model capacity with training (See Appendix. F). In the experiments, we refer to this structure as the shortcut (Sc.) branch and the separate BN (SepBN.) branch.

This structure is also inspired by the result of Li et al. (2020a), where it is crucial to use individual batch normalization parameters for the small and the big view to achieve valid improvements. However, it doubles the computation for the convolutions to additionally process the big view, since it performs aggregation before convolutions. In comparison, our aggregation module is placed after the TAdaConv, which is thus exempted from the doubled computation for the preceding convolutions and meanwhile being able to effectively aggregate information along the temporal dimension.

5 Experiments on video classification

To show the effectiveness and generality of the proposed approach, we present comprehensive evaluation of TAdaConv and TAda2D on two video understanding tasks using four large-scale datasets.

Datasets. For video classification, we use Kinetics-400 (Kay et al., 2017), Something-Something-V2 (Goyal et al., 2017), and Epic-Kitchens-100 (Damen et al., 2020). *K400* is a widely used action classification dataset with 400 categories covered by ∼300K videos. *SSV2* includes 220K videos with challenging spatio-temporal interactions in 174 classes. *EK100* includes 90K action segments labelled by 97 verb and 300 noun classes, which requires more comprehensive understanding of videos as the model need to predict both verb and noun correctly. For temporal action localization (Sec. 6), besides EK100, we employ HACS (Zhao et al., 2019), which contains 139K action segments from 50K untrimmed videos covering 200 classes.

Training and evaluation. For training TAda2D, 8 or 16 frames are sampled by interval (K400&EK100) or by segment (SSV2) with temporal jittering, following convention (Lin et al., 2019a; Liu et al., 2021; Feichtenhofer et al., 2019). A spatial region of 224×224 is cropped from frames with shorter side resized to a random value in [256, 320]. Further training details can be found in the appendix A. For evaluation, we use three spatial crops with 10 clips (K400&EK100) or 2 clips (SSV2) uniformly sampled along the temporal dimension. Each crop has the size of 256×256, which is obtained from a video with its shorter side resized to 256.

Table 2: K400 and SSV2 classification performance with and without TAdaConv. Notation $\star$ inicates our own implementation. See Appendix B for details on the model structure.

Base Model	TAdaConv	Frames	Params.	GFLOPs	K400	Δ	SSV2	Δ
SlowOnly $8\times8^\star$	✗	8	32.5M	54.52	74.56	-	60.31	-
	✓	8	35.6M	54.53	75.85	**+1.29**	63.30	**+2.99**
SlowFast $4\times16^\star$	✗	4+32	34.5M	36.10	75.03	-	56.71	-
	✓	4+32	37.7M	36.11	76.47	**+1.44**	59.80	**+3.09**
SlowFast $8\times8^\star$	✗	8+32	34.5M	65.71	76.19	-	61.54	-
	✓	8+32	37.7M	65.73	77.43	**+1.24**	63.88	**+2.34**
R(2+1)D*	✗	8	28.1M	49.55	73.63	-	61.06	-
	✓(2d)	8	30.0M	49.57	75.19	**+1.56**	62.86	**+1.80**
	✓(2d+1d)	8	31.9M	49.58	75.36	**+1.73**	63.78	**+2.72**
R3D*	✗	8	46.5M	84.23	73.83	-	59.86	-
	✓(3d)	8	48.5M	84.24	74.91	**+1.08**	62.85	**+2.99**

Table 3: Ablation studies.

(a) **Calibration weight generation.** *K:* kernel size; *Lin./Non-Lin.*: linear/non-linear weight generation; *G:* global information.

Model	TAdaConv	K.	G.	Top-1
TSN*	-	-	-	32.0
Ours	Lin.	1	✗	37.5
	Lin.	3	✗	56.5
	Non-Lin.	(1, 1)	✗	36.8
	Non-Lin.	(3, 1)	✗	57.1
	Non-Lin.	(1, 3)	✗	57.3
	Non-Lin.	(3, 3)	✗	57.8
	Lin.	1	✓	53.4
	Non-Lin.	(1, 1)	✓	54.4
	Non-Lin.	(3, 3)	✓	59.2

(b) **Feature aggregation scheme.** *FA:* feature aggregation; *Sc:* shortcut for convolution feature; *SepBN:* separate batch norm.

TAdaConv	FA.	Sc.	SepBN.	Top-1	Δ
✗	-	-	-	32.0	-
✓	-	-	-	59.2	+27.2
✗	Avg.	✗	-	47.9	+15.9
✗	Avg.	✓	✗	49.0	+17.0
✗	Avg.	✓	✓	57.0	+25.0
✓	Avg.	✗	-	60.1	+28.1
✓	Avg.	✓	✗	61.5	+29.5
✓	Avg.	✓	✓	63.8	**+31.8**
✓	Max.	✓	✓	63.5	+31.5
✓	Mix.	✓	✓	63.7	+31.7

5.1 TAdaConv on existing video backbones

TAdaConv is designed as a plug-in substitution for the spatial convolutions in the video models. Hence, we first present plug-in evaluations in Table 2. TAdaConv is observed to bring improvement on the classification accuracy with negligible computation overhead on a wide range of video models, including SlowFast (Feichtenhofer et al., 2019), R3D (Hara et al., 2018) and R(2+1)D (Tran et al., 2018), with an average improvement of 1.3% and 2.8% respectively on K400 and SSV2 at an extra computational cost of less than 0.02GFlops. These models take advantage of different strategies for temporal modelling, including slowfast branches, temporal convolutions, and 3D convolutions. We show that TAdaConv can not only improve spatial convolutions, but also 3D convolutions or temporal convolutions. For fair comparison, all models are trained using the same training strategy. Further plug-in evaluation on the action classification task is presented in Appendix D.

5.2 Ablation studies and main results on Something-Something-V2

We present thorough ablation studies for the justification of our design choices and the effectiveness of our TAdaConv in modelling temporal dynamics. SSV2 is used as the evaluation benchmark, as it is widely acknowledged to have more complex spatio-temporal interactions.

Calibration weight generation function. We ablate different ways to generate the calibration weight in Table 3a. Our baseline is a 2D model, which corresponds to the structure of temporal segment network (Wang et al., 2016) (See Appendix B for more details). When using linear weight generation functions, TAdaConv without local temporal context can improve the baseline by 5.5%. On top of this, enlarging the kernel of the 1D convolution in the weight generation process yields a further improvement of 19%, which demonstrates the benefit of weight calibration and the importance of the local temporal context. A stronger non-linear weight generation further improves the performance, especially when the local temporal context is included. Finally, for the scope of tempo-

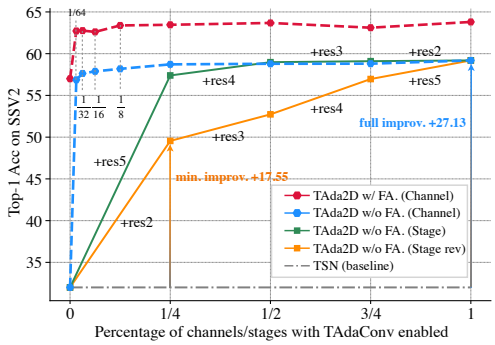


Figure 3: The classification performance of TAda2D consistently increases when the percentage of channels or the number of stages with TAdaConv enabled increases.

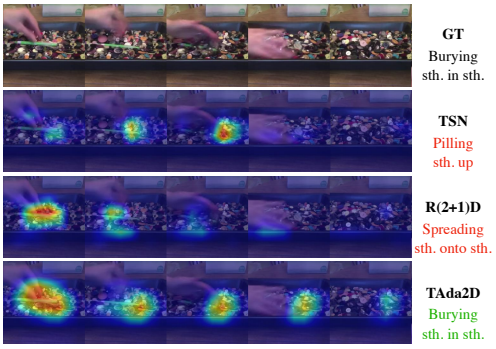


Figure 4: Comparison of the Grad-CAM (Selvaraju et al., 2017) visualization and prediction between TSN, R(2+1)D and our TAda2D. See Fig. A3 for more examples.

ral context, although global temporal context provides less accurate weight calibrations, combining the global and local temporal context yields a better performance for both variants.

Feature aggregation. The feature aggregation module used in the TAda2D network is ablated in Table 3b. First, the performance is similar for plain aggregation $x = \text{Avg}(x)$ and aggregation with a shortcut (Sc.) branch $x = x + \text{Avg}(x)$, with Sc. being slightly better. Separating the batchnorm (Eq. 7) for the shortcut and the aggregation branch brings notable improvement. Max pooling and mix pooling (avg+max) slightly underperform the average pooling variant. Overall, the combination of TAdaConv and our feature aggregation scheme has an advantage over the TSN baseline of 31.8%.

Calibration dimension. As discussed, there are multiple dimensions in the base weight that can be calibrated. Table 4 shows that calibrating the channel dimension is a better option for calibrating the spatial dimension, which means that the spatial structure of the original convolution kernel should be retained. The calibration on the C_{in} dimension performs better than calibrating C_{out} or both combined. This is probably because the calibration weight is generated by the input feature, which makes the calibrated convolution weight better suited to the input feature to be further applied on the feature.

Table 4: **Calibration dimension.**

Cal. dim.	Δ Parms.	Δ GFLOPs	Top-1
C_{in}	3.16M	0.016	63.8
C_{out}	3.16M	0.016	63.4
$C_{\text{in}} \times C_{\text{out}}$	4.10M	0.024	63.7
k^2	2.24M	0.009	62.7

Different stages employing TAdaConv. The solid lines in Fig 3 show the stage by stage replacement of the spatial convolutions in a ResNet model. It has a minimum improvement of 17.55%, when TAdaConv is employed in *Res2*. Compared to early stages, later stages contribute more to the final performance, as later stages provide more accurate calibration because of its high abstraction level. Overall, TAdaConv is used in all stages for the highest accuracy.

Different proportion of channels employing TAdaConv. Here, we calibrate only a proportion of channels using TAdaConv and leave the other channels uncalibrated. The results are presented as dotted lines in Fig 3. We find temporally adaptive weight calibration can improve the baseline by a large margin even if only 1/64 channels are calibrated. The performance is further improved when the proportion is further enlarged, but less significantly compared with from no TAdaConv to enabling it in 1/64 channels.

Visualizations. We qualitatively evaluate our approach in comparison with the baseline approaches (TSN and R(2+1)D) by presenting the Grad-CAM (Selvaraju et al., 2017) visualizations of the last stage in Fig. 4. TAda2D demonstrates a stronger ability in spotting complete key information in the video, thanks to the temporal reasoning based on global spatial information as well as the additional global temporal information.

Main results. In comparison with the state-of-the-arts, the main results on SSV2 is listed in Table 5. TAda2D outperforms previous approaches using the same number of frames. Compared to the video-level model TDN where more frames are used, TAda2D performs competitively.

Table 5: Comparison with the top approaches on Something-Something-V2 (Goyal et al., 2017). Gray font indicate models with more frames as inputs.

Model	Backbone	Frames×clips×crops	Top-1	Top-5
TDN (Wang et al., 2021)	ResNet-50	$(8f+32f)\times1\times1$	64.0	88.8
TDN (Wang et al., 2021)	ResNet-50	$(16f+64f)\times1\times1$	65.3	89.5
TDN (Wang et al., 2021)	ResNet-50	$(8f+32f+16f+64f)\times1\times1$	67.0	90.3
TSM (Lin et al., 2019a)	ResNet-50	$8f\times2\times3$	59.1	85.6
TSM (Lin et al., 2019a)	ResNet-50	$16f\times2\times3$	63.4	88.5
STM (Jiang et al., 2019)	ResNet-50	$16f\times10\times3$	64.2	89.8
SmallBigNet (Li et al., 2020a)	ResNet-50	$8f\times2\times3$	61.6	87.7
SmallBigNet (Li et al., 2020a)	ResNet-50	$16f\times2\times3$	63.8	88.9
SmallBigNet (Li et al., 2020a)	ResNet-50	$(8f+16f)\times2\times3$	64.5	89.1
TANet (Liu et al., 2021)	ResNet-50	$8f\times2\times3$	62.7	88.0
TANet (Liu et al., 2021)	ResNet-50	$16f\times2\times3$	64.6	89.5
TEA (Li et al., 2020b)	Res2Net-50	$16f\times10\times3$	65.1	89.9
TAda2D (Ours)	ResNet-50	$8f\times2\times3$	63.8	87.7
TAda2D (Ours)	ResNet-50	$16f\times2\times3$	65.2	89.1
TAda2D$_{\text{En}}$ (Ours)	ResNet-50	$(8f+16f)\times2\times3$	66.7	89.5

Table 6: Comparison with the state-of-the-art approaches over action classification on Epic-Kitchens-100 (Damen et al., 2020). $\star$ indicates our own implementation for fair comparison. $\uparrow$ indicates the main evaluation metric for the dataset.

		Top-1			Top-5		
Model	Frames	Act.↑	Verb	Noun	Act.↑	Verb	Noun
TSN (Wang et al., 2016)	8	33.19	60.18	46.03	55.13	89.59	72.90
TRN (Zhou et al., 2018)	8	35.34	65.88	45.43	56.74	90.42	71.88
TSM (Lin et al., 2019a)	8	38.27	67.86	49.01	60.41	90.98	74.97
SlowFast (Feichtenhofer et al., 2019)	8+32	38.54	65.56	50.02	58.60	90.00	75.62
TSN* (Our baseline)	8	29.25	51.51	45.07	52.59	87.95	71.27
TAda2D (Ours)	8	40.89	64.57	51.90	59.22	90.16	76.59

5.3 Main results on Kinetics and Epic-Kitchens

Epic-Kitchens-100. Table 6 lists our results on EK100 in comparison with the previous approaches[3]. We calculate the final action prediction following the strategies in Huang et al. (2021). For fair comparison, we reimplemented our baseline TSN using the same training and evaluation strategies. TAda2D improves this baseline by 11.64% on the action prediction. Over previous approaches, TAda2D achieves a higher accuracy with a notable margin.

Kinetics-400. Comparison with the state-of-the-art models on the Kinetics-400 benchmark is presented in Table 7, where we show TAda2D performs competitively against previous state-of-the-arts. In comparison with the models using the similar backbone and the same number of frames, TAda2D achieves competitive accuracy. The ensemble of TAda2D using 8 and 16 frames achieves a higher accuracy than some of the previous works using more frames, *e.g.,* 77.5 for TDN using (16+64) frames vs 77.9 for ours using 24 frames.

6 Experiments on temporal action localization

Dataset and evaluation metrics. Action localization is an essential task for understanding untrimmed videos. Current action localization pipeline relies heavily on the quality of the video representations, which puts high requirements on the generality of the video models. The generality of our TAda2D is evaluated on two large-scale action localization datasets, HACS (Zhao et al., 2019) and Epic-Kitchens-100 (Damen et al., 2020). For evaluation, we mainly follow the protocol in the respective dataset: the average mean Average Precision (average mAP) at IoU threshold [0.5:0.05:0.95] for HACS and [0.1:0.1:0.5] for Epic-Kitchens-100.

[3]The performances are referenced from the official release of the EK100 dataset (Damen et al., 2020).

Table 7: Comparison with the state-of-the-art approaches on Kinetics 400 (Kay et al., 2017). The $\star$ indicates the FLOPs is calculated with a spatial resolution of 224×224, while the performance is obtained with 256×256. The actual computation can be calculated by a multiplication factor of 1.306 (*e.g.*, 33×1.306=43.1). Gray font indicates models with different backbone or inputs.

Model	Backbone	Frames	GFLOPs	Top-1	Top-5
I3D (Carreira & Zisserman, 2017)	Inception	64	108×N/A	72.1	90.3
S3D-G (Xie et al., 2018)	Inception	64	71.4×N/A	74.7	93.4
X3D-M (Feichtenhofer, 2020)	X3D	16	6.2×30	76.0	92.3
X3D-L (Feichtenhofer, 2020)	X3D	16	24.8×30	77.5	92.9
R(2+1)D (Tran et al., 2018)	ResNet-34	32	152×10	74.3	91.4
TEA (Li et al., 2020b)	Res2Net-50	8	35×30*	75.0	91.8
TEA (Li et al., 2020b)	Res2Net-50	16	70×30*	76.1	92.5
TDN (Wang et al., 2021)	ResNet-50	8+32	36×30*	76.6	92.8
TDN (Wang et al., 2021)	ResNet-50	16+64	72×30*	77.5	93.2
SlowFast 4×16 (Feichtenhofer et al., 2019)	ResNet-50	4+32	36.1×30	75.6	92.1
SlowFast 8×8 (Feichtenhofer et al., 2019)	ResNet-50	8+32	65.7×30	77.0	92.6
TSM (Lin et al., 2019a)	ResNet-50	8	33×30*	74.1	N/A
NL I3D (Wang et al., 2018a)	ResNet-50	32	N/A×N/A	74.9	91.6
NL I3D (Wang et al., 2018a)	ResNet-50	128	282×30	76.5	92.6
SmallBigNet (Li et al., 2020a)	ResNet-50	8	57×30*	76.3	92.5
TANet (Liu et al., 2021)	ResNet-50	8	43×30	76.3	92.6
TANet (Liu et al., 2021)	ResNet-50	16	86×30	76.9	92.9
CorrNet (Wang et al., 2020)	ResNet-50	32	115×10	77.2	N/A
TAda2D (Ours)	ResNet-50	8	43.1×30	76.3	92.4
TAda2D (Ours)	ResNet-50	16	86.2×30	76.9	92.7
TAda2D$_{\text{En}}$ (Ours)	ResNet-50	8+16	129.3×30	77.9	93.1

Table 8: Action localization evaluation on HACS and Epic-Kitchens-100. ↑ indicates the main evaluation metric for the dataset, *i.e.,* average mAP for action localization.

	HACS						Epic-Kitchen-100						
Model	@0.5	@0.6	@0.7	@0.8	@0.9	**Avg.**↑	Task	@0.1	@0.2	@0.3	@0.4	@0.5	**Avg.**↑
TSN	43.6	37.7	31.9	24.6	15.0	28.6	Verb	15.98	15.01	14.09	12.25	10.01	13.47
							Noun	15.11	14.15	12.78	10.94	8.89	12.37
							Act.↑	10.24	9.61	8.94	7.96	6.79	8.71
TAda2D	48.7	42.7	36.2	28.1	17.3	32.3	Verb	19.70	18.49	17.41	15.50	12.78	16.78
							Noun	20.54	19.32	17.94	15.77	13.39	17.39
							Act.↑	15.15	14.32	13.59	12.18	10.65	13.18

Localization pipeline. Following previous works (Damen et al., 2020; Qing et al., 2021a;c), the localization pipeline generally follows three procedures: **(i)** finetuning the model on the respective dataset; **(ii)** training BMN (Lin et al., 2019b) for action proposals using features of finetuned model; **(iii)** classifying proposals to obtain final results. We include further details in Appendix A.

Main results. Table 8 compares the localization result of TAda2D with our reimplemented baseline, which shows that TAda2D also provides a stronger feature for detecting actions along the temporal dimension. We further perform plug-in evaluation on action localization in Appendix E, showing that TAdaConv can also improve existing feature extractors on the localization task.

7 Conclusions

This work proposes Temproally-Adaptive Convolutions (TAdaConv) for video understanding, which calibrates the convolution weights for each frame based on its local and global temporal context in a video. TAdaConv demonstrates superior temporal modelling abilities both quantitatively and qualitatively on both video classification and temporal localization tasks. TAdaConv can serve both as a strong stand-alone temporal modelling module in 2D backbones, and as the plug-in replacement in video backbones to further enhance the model capacity. We hope this work can facilitate further research in video understanding.

REFERENCES

Sergey Bartunov, Adam Santoro, Blake A Richards, Luke Marris, Geoffrey E Hinton, and Timothy P Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 9390–9400, 2018.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

Jin Chen, Xijun Wang, Zichao Guo, Xiangyu Zhang, and Jian Sun. Dynamic region-aware convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8064–8073, 2021.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4): 834–848, 2017.

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11030–11039, 2020a.

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic relu. In *European Conference on Computer Vision*, pp. 351–367. Springer, 2020b.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 764–773, 2017.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Rescaling egocentric vision. *arXiv preprint arXiv:2006.13256*, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Gamaleldin Elsayed, Prajit Ramachandran, Jonathon Shlens, and Simon Kornblith. Revisiting spatial invariance with low-rank local connectivity. In *International Conference on Machine Learning*, pp. 2868–2879. PMLR, 2020.

Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 203–213, 2020.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, pp. 6202–6211, 2019.

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 *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, volume 1, pp. 5, 2017.

Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 6546–6555, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Ziyuan Huang, Zhiwu Qing, Xiang Wang, Yutong Feng, Shiwei Zhang, Jianwen Jiang, Zhurong Xia, Mingqian Tang, Nong Sang, and Marcelo H Ang Jr. Towards training stronger video vision transformers for epic-kitchens-100 action recognition. *arXiv preprint arXiv:2106.05058*, 2021.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29:667–675, 2016.

Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2000–2009, 2019.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 510–519, 2019.

Xianhang Li, Yali Wang, Zhipeng Zhou, and Yu Qiao. Smallbignet: Integrating core and contextual views for video classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1092–1101, 2020a.

Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 909–918, 2020b.

Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning dynamic routing for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8553–8562, 2020c.

Yunsheng Li, Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Lu Yuan, Zicheng Liu, Lei Zhang, and Nuno Vasconcelos. Micronet: Towards image recognition with extremely low flops. *arXiv preprint arXiv:2011.12289*, 2020d.

Yunsheng Li, Yinpeng Chen, Xiyang Dai, Dongdong Chen, Ye Yu, Lu Yuan, Zicheng Liu, Mei Chen, Nuno Vasconcelos, et al. Revisiting dynamic convolution via matrix decomposition. In *International Conference on Learning Representations*, 2021.

Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7083–7093, 2019a.

Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. Bmn: Boundary-matching network for temporal action proposal generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3889–3898, 2019b.

Zhaoyang Liu, Limin Wang, Wayne Wu, Chen Qian, and Tong Lu. Tam: Temporal adaptive module for video recognition. *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Yue Meng, Rameswar Panda, Chung-Ching Lin, Prasanna Sattigeri, Leonid Karlinsky, Kate Saenko, Aude Oliva, and Rogerio Feris. Adafuse: Adaptive temporal fusion network for efficient action recognition. In *International Conference on Learning Representations*, 2021.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

Bruno A Olshausen and David J Field. Natural image statistics and efficient coding. *Network: computation in neural systems*, 7(2):333, 1996.

Zhiwu Qing, Ziyuan Huang, Xiang Wang, Yutong Feng, Shiwei Zhang, Jianwen Jiang, Mingqian Tang, Changxin Gao, Marcelo H Ang Jr, and Nong Sang. A stronger baseline for ego-centric action detection. *arXiv preprint arXiv:2106.06942*, 2021a.

Zhiwu Qing, Haisheng Su, Weihao Gan, Dongliang Wang, Wei Wu, Xiang Wang, Yu Qiao, Junjie Yan, Changxin Gao, and Nong Sang. Temporal context aggregation network for temporal action proposal refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 485–494, 2021b.

Zhiwu Qing, Xiang Wang, Ziyuan Huang, Yutong Feng, Shiwei Zhang, Mingqian Tang, Changxin Gao, Nong Sang, et al. Exploring stronger feature for temporal action localization. *arXiv preprint arXiv:2106.13014*, 2021c.

Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pp. 5533–5541, 2017.

Daniel L Ruderman and William Bialek. Statistics of natural images: Scaling in the woods. *Physical review letters*, 73(6):814, 1994.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Eero P Simoncelli and Bruno A Olshausen. Natural image statistics and neural representation. *Annual review of neuroscience*, 24(1):1193–1216, 2001.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497, 2015.

Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 6450–6459, 2018.

Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5552–5561, 2019.

Heng Wang, Du Tran, Lorenzo Torresani, and Matt Feiszli. Video modeling with correlation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 352–361, 2020.

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 *European conference on computer vision*, pp. 20–36. Springer, 2016.

Limin Wang, Zhan Tong, Bin Ji, and Gangshan Wu. Tdn: Temporal difference networks for efficient action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1895–1904, 2021.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7794–7803, 2018a.

Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 409–424, 2018b.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 305–321, 2018.

Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *arXiv preprint arXiv:1904.04971*, 2019.

Hang Zhao, Antonio Torralba, Lorenzo Torresani, and Zhicheng Yan. Hacs: Human action clips and segments dataset for recognition and temporal localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8668–8678, 2019.

Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 803–818, 2018.

Daquan Zhou, Xiaojie Jin, Qibin Hou, Kaixin Wang, Jianchao Yang, and Jiashi Feng. Neural epitome search for architecture-agnostic network compression. *arXiv preprint arXiv:1907.05642*, 2019.

APPENDIX

In the appendix, we provide further implementation details (Appendix A) on the action classification and localization, model structures that we used for evaluation (Appendix B), per-category improvement analysis on Something-Something-V2 (Appendix C), further plug-in evaluations on Epic-Kitchens classification (Appendix D) plug-in evaluations on the temporal action localization task (Appendix E), as well as the visualization of the training procedure of TSN and TAda2D (Appendix F). Further, we show additional qualitative analysis in Fig. A3.

A FURTHER IMPLEMENTATION DETAILS

Here, we further describe the implementation details for the action classification and action localization experiments. For fair comparison, we keep all the training strategies the same for our baseline, the plug-in evaluations as well as our own models.

A.1 ACTION CLASSIFICATION

Our experiments on the action classification are conducted on three large-scale datasets. For all action classification models, we train them with synchronized SGD using 16 GPUs. The batch size for each GPU is 16 and 8 respectively for 8-frame and 16-frame models. The weights in TAda2D are initialized using ImageNet (Deng et al., 2009) pre-trained weights (He et al., 2016), except for the calibration function $\mathcal{G}$ and the batchnorm statistics (BN_2) in the average pooling branch. In the calibration function, we randomly initialize the first convolution layer (for non-linear weight generation) following He et al. (2015), and fill zero to the weight of last convolution layer. The batchnorm statistics are initialized to be zero so that the initial state behaves the same as without the average pooling branch. For all models, we use a dropout ratio (Hinton et al., 2012) of 0.5 before the classification heads.

On *Kinetics-400*, a half-period cosine schedule is applied for decaying the learning rate following Feichtenhofer et al. (2019), with the base learning rate set to 0.24. The models are trained for 100 epochs. In the first 8 epochs, we adopt a linear warm-up strategy starting from a learning rate of 0.01. The weight decay is set to 1e-4. The frames are sampled based on a fixed interval, which is 8 for 8-frame models and 5 for 16-frame models.

On *Epic-Kitchens-100*, the models are initialized with weights pre-trained on Kinetics-400, and are further fine-tuned following a similar strategy as in kinetics. The training length is reduced to 50 epochs, with 10 epochs for warm-up. The base learning rate is 0.48. Following Damen et al. (2020) and Huang et al. (2021), we connect two separate heads for predicting verbs and nouns. Action predictions are obtained according to the strategies in Huang et al. (2021), which is shown to have a higher accuracy over the original one in Damen et al. (2020). For fair comparison, we also trained and evaluated our baseline using the same strategy.

On *Something-Something-V2*, we initialize the model with ImageNet pretrained weights. A segment-based sampling strategy is adopted, where for T-frame models, the video is divided into T segments before one frame is sampled from each segment randomly for training or uniformly for evaluation. The models are trained for 64 epochs, with the first 4 being the warm-up epochs. The base learning rate is set to 0.48.

It is worth noting that, for SlowFast models (Feichtenhofer et al., 2019) in the plug-in evaluations, we do not apply precise batch normalization statistics in our implementation as in its open-sourced codes, which is possibly the reason why our re-implemented performance is slightly lower than the original published numbers.

A.2 ACTION LOCALIZATION

We evaluate our model on the action localization task using two large-scale datasets. The overall pipeline for our action localization evaluation is divided into finetuning the classification models, obtaining action proposals and classifying the proposals.

Table A1: Model structure of R3D, R(2+1)D and R2D that we used in our experiments. **Blue** and **green** fonts indicate respectively the default convolution operation and optional operation that can be replaced by TAdaConv. (Better viewed in color.)

Stage	R3D	R(2+1)D	R2D (default baseline)	output sizes
sampling	interval 8, 1^2	interval 8, 1^2	interval 8, 1^2	$8\times224\times224$
conv_1	3×7^2, 64 stride 1, 2^2	1×7^2, 64 stride 1, 2^2	1×7^2, 64 stride 1, 2^2	$8\times112\times112$
res_2	$\begin{bmatrix} 1\times1^2, 64 \\ \mathbf{3\times3^2, 64} \\ 1\times1^2, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1^2, 64 \\ \mathbf{1\times3^2, 64} \\ \mathbf{3\times1^2, 64} \\ 1\times1^2, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1^2, 64 \\ \mathbf{1\times3^2, 64} \\ 1\times1^2, 256 \end{bmatrix}\times3$	$8\times56\times56$
res_3	$\begin{bmatrix} 1\times1^2, 128 \\ \mathbf{3\times3^2, 128} \\ 1\times1^2, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1^2, 128 \\ \mathbf{1\times3^2, 128} \\ \mathbf{3\times1^2, 128} \\ 1\times1^2, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1^2, 128 \\ \mathbf{1\times3^2, 128} \\ 1\times1^2, 512 \end{bmatrix}\times4$	$8\times28\times28$
res_4	$\begin{bmatrix} 1\times1^2, 256 \\ \mathbf{3\times3^2, 256} \\ 1\times1^2, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1^2, 256 \\ \mathbf{1\times3^2, 256} \\ \mathbf{3\times1^2, 256} \\ 1\times1^2, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1^2, 256 \\ \mathbf{1\times3^2, 256} \\ 1\times1^2, 1024 \end{bmatrix}\times6$	$8\times14\times14$
res_5	$\begin{bmatrix} 1\times1^2, 512 \\ \mathbf{3\times3^2, 512} \\ 1\times1^2, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1^2, 512 \\ \mathbf{1\times3^2, 512} \\ \mathbf{3\times1^2, 512} \\ 1\times1^2, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1^2, 512 \\ \mathbf{1\times3^2, 512} \\ 1\times1^2, 2048 \end{bmatrix}\times3$	$8\times7\times7$
global average pool, fc				$1\times1\times1$

Finetuning. On *Epic-Kitchens*, we simply use the evaluated action classification model. On *HACS*, following (Qing et al., 2021c), we initialize the model with Kinetics-400 pre-trained weights and use adamw (Loshchilov & Hutter, 2017) to train the model for 30 epochs with 8 epochs for warm-up using 32 GPUs. The mini-batch size is 16 videos per GPU. The base learning rate is set to 0.0002, with cosine learning rate decay as in Kinetics. In our case, only the segments with action labels are used for training.

Proposal generation. For the action proposals, a boundary matching network (BMN) (Lin et al., 2019b) is trained over the extracted features on the two datasets. On *Epic-Kitchens*, we extract features with the videos uniformly decoded at 60 FPS. For each clip, we use 8 frames with an interval of 8 to be consistent with finetuning, which means a feature roughly covers a video clip of one seconds. The interval between each clip for feature extraction is 8 frames (*i.e.,* 0.133 sec) as well. The shorter side of the video is resized to 224 and we feed the whole spatial region into the backbone to retain as much information as possible. Following Qing et al. (2021a), we generate proposals using BMN based on sliding windows. The predictions on the overlapped region of different sliding windows are simply averaged. On *HACS*, the videos are decoded at 30 FPS, and extend the interval between clips to be 16 (*i.e.,* 0.533 sec) because the actions in HACS last much longer than in Epic-Kitchens. The shorter side is resized to 128 for efficient processing. For the settings in generating proposals, we mainly follow Qing et al. (2021c), except that the temporal resolution is resized to 100 in our case instead of 200.

Classification. On *Epic-Kitchens*, we classify the proposals with the fine-tuned model using 6 clips. Spatially, to comply with the feature extraction process, we resize the shorter side to 224 and feed the whole spatial region to the model for classification. On *HACS*, considering the property of the dataset that only one action category can exist in a video, we obtain the video level classification results by classifying the video level features, following Qing et al. (2021c).

B Model Structures

The detailed model structures for R2D, R(2+1)D and R3D is specified in Table A1. We highlight the convolutions that are replaced by TAdaConv by default or optionally. For all of our models, a small modification is made in that we remove the max pooling layer after the first convolution and set the

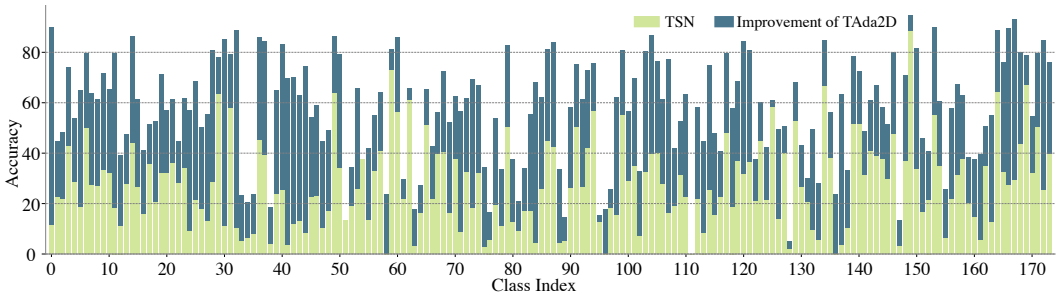


Figure A1: **Per-category performance comparison of TAda2D against the baseline TSN.** We achieve an average per-category performance improvement of 30.35%.

Table A2: Comparison with the state-of-the-art approaches over action classification on Epic-Kitchens-100 (Damen et al., 2020). ↑ indicates the main evaluation metric for the dataset. For fair comparison, we implement all the baseline models using our own training strategies.

		Top-1			**Top-5**		
Model	**Frames**	**Act.↑**	Verb	Noun	**Act.↑**	Verb	Noun
SlowFast 4×16	4+32	38.17	63.54	48.79	58.68	89.75	73.37
SlowFast 4×16 + TAdaConv	4+32	39.14	64.50	49.59	59.21	89.67	73.88
SlowFast 8×8	8+32	40.08	65.05	50.72	60.10	90.04	74.26
SlowFast 8×8 + TAdaConv	8+32	41.35	66.36	52.32	61.68	90.59	75.89
R(2+1)D	8	37.45	62.92	48.27	58.02	89.75	73.60
R(2+1)D + TAdaConv$_{2d}$	8	39.72	64.48	50.26	60.22	90.01	75.06
R(2+1)D + TAdaConv$_{2d+1d}$	8	40.10	64.77	50.28	60.45	89.99	75.55
R3D	8	36.67	61.92	47.87	57.47	89.02	73.05
R3D + TAdaConv$_{3d}$	8	39.30	64.03	49.94	59.67	89.84	74.56

spatial stride of the second stage to be 2, following Wang et al. (2020). Temporal resolution is kept unchanged following recent works (Feichtenhofer et al., 2019; Li et al., 2020b; Jiang et al., 2019). Our *R3D* is obtained by simply expanding the R2D baseline in the temporal dimension by a factor of three. We initialize with weights reduced by 3 times, which means the original weight is evenly distributed in adjacent time steps. We construct the *R(2+1)D* by adding a temporal convolution operation after the spatial convolution. The temporal convolution can also be optionally replaced by TAdaConv, as shown in Table 2 and Table A2. For its initialization, the temporal convolution weights are randomly initialized, while the others are initialized with the pre-trained weights on ImageNet. For SlowFast models, we keep all the model structures identical to the original work (Feichtenhofer et al., 2019).

C Per-category Improvement Analysis on SSV2

This section provides a per-category improvement analysis on the Something-Something-V2 dataset in Fig.A1. As shown in Table 3b, our TAda2D achieves an overall improvement of 31.7%. Our per-category analysis shows an mean improvement of 30.35% over all the classes. The largest improvement is observed in class 0 (78.5%, *Approaching something with your camera*), 32 (78.4%, *Moving away from something with your camera*), 30 (74.3%, *Lifting up one end of something without letting it drop down*), 44 (66.2%, *Moving something towards the camera*) and 41 (66.1%, *Moving something away from the camera*). Most of these categories contain large movements across the whole video, whose improvement benefits from temporal reasoning over the global spatial context. For class 30, most of its actions lasts a long time (as it needs to be determined whether the end of something is let down or not). The improvements over the baseline mostly benefits from the global temporal context that are included in the weight generation process.

Table A3: Plug-in evaluation of TAdaConv on the action localization on HACS and Epic-Kitchens. ↑ indicates the main evaluation metric for the dataset.

	HACS						Epic-Kitchen-100						
Model	@0.5	@0.6	@0.7	@0.8	@0.9	**Avg.**↑	Task	@0.1	@0.2	@0.3	@0.4	@0.5	**Avg.**↑
							Verb	19.93	18.92	17.90	16.08	13.24	17.21
SlowFast 8×8	50.0	44.1	37.7	29.6	18.4	33.7	Noun	17.93	16.83	15.53	13.68	11.41	15.07
							Act.↑	14.00	13.19	12.37	11.18	9.52	12.04
							Verb	19.96	18.71	17.65	15.41	13.35	17.01
SlowFast 8×8 + TAdaConv	51.7	45.7	39.3	31.0	19.5	35.1	Noun	20.17	18.90	17.58	15.83	13.18	17.13
							Act.↑	14.90	14.12	13.32	12.07	10.57	13.00

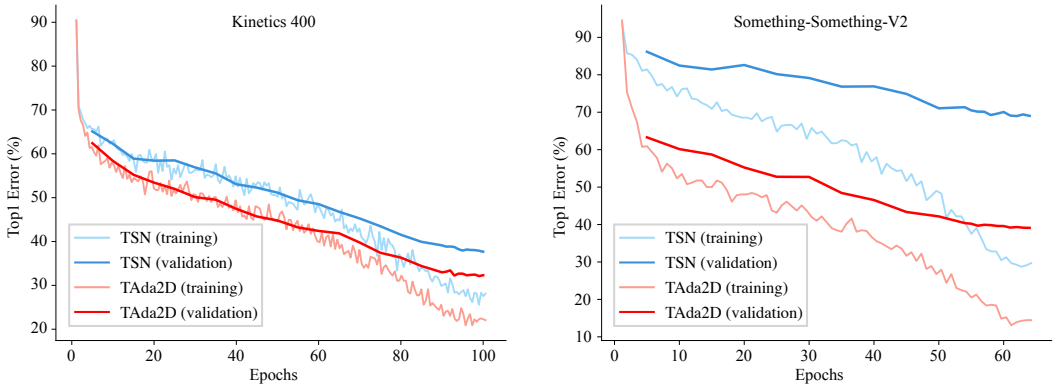


Figure A2: **Training and validation on Kinetics-400 and Something-Something-V2.** On both datasets, TAda2D shows a stronger capability of fitting the data and a better generality to the validation set. Further, TAda2D reduces the overfitting problem in Something-Something-V2.

D Further plug-in evaluation for TAdaConv on classification

In complement to Table 2, we further show in Table A2 the plug-in evaluation on the action classification task on the Epic-Kitchens-100 dataset. As in the plug-in evaluation on Kinetics and Something-Something-V2, we compare performances with and without TAdaConv over three baseline models, SlowFast (Feichtenhofer et al., 2019), R(2+1)D (Tran et al., 2018) and R3D (Hara et al., 2018) respectively representing three kinds of temporal modeling techniques. The results are in line with our observation in Table 2. Over all three kinds of temporal modelling strategies, adding TAdaConv further improves the recognition accuracy of the model.

E Plug-in evaluation for TAdaConv on Action Localization

Here, we show the plug-in evaluation on the temporal action localization task. Specifically, we use SlowFast as our baseline, as it is shown to be superior in the localization performance in Qing et al. (2021b) compared to many early backbones. The result is presented in Table A3. With TAdaConv, the average mAP on HACS is improved by 1.4%, and the average mAP on Epic-Kitchens-100 action localization is improved by 1.0%.

F Comparison of training procedure

In this section, we compare the training procedure of TSN and TAda2D on Kinetics-400 and Something-Something-V2. The results are presented in Fig. A2. TAda2D demonstrates a stronger fitting ability and generality even from the early stages of the training, despite that the initial state of TAda2D is identical to that of TSN.

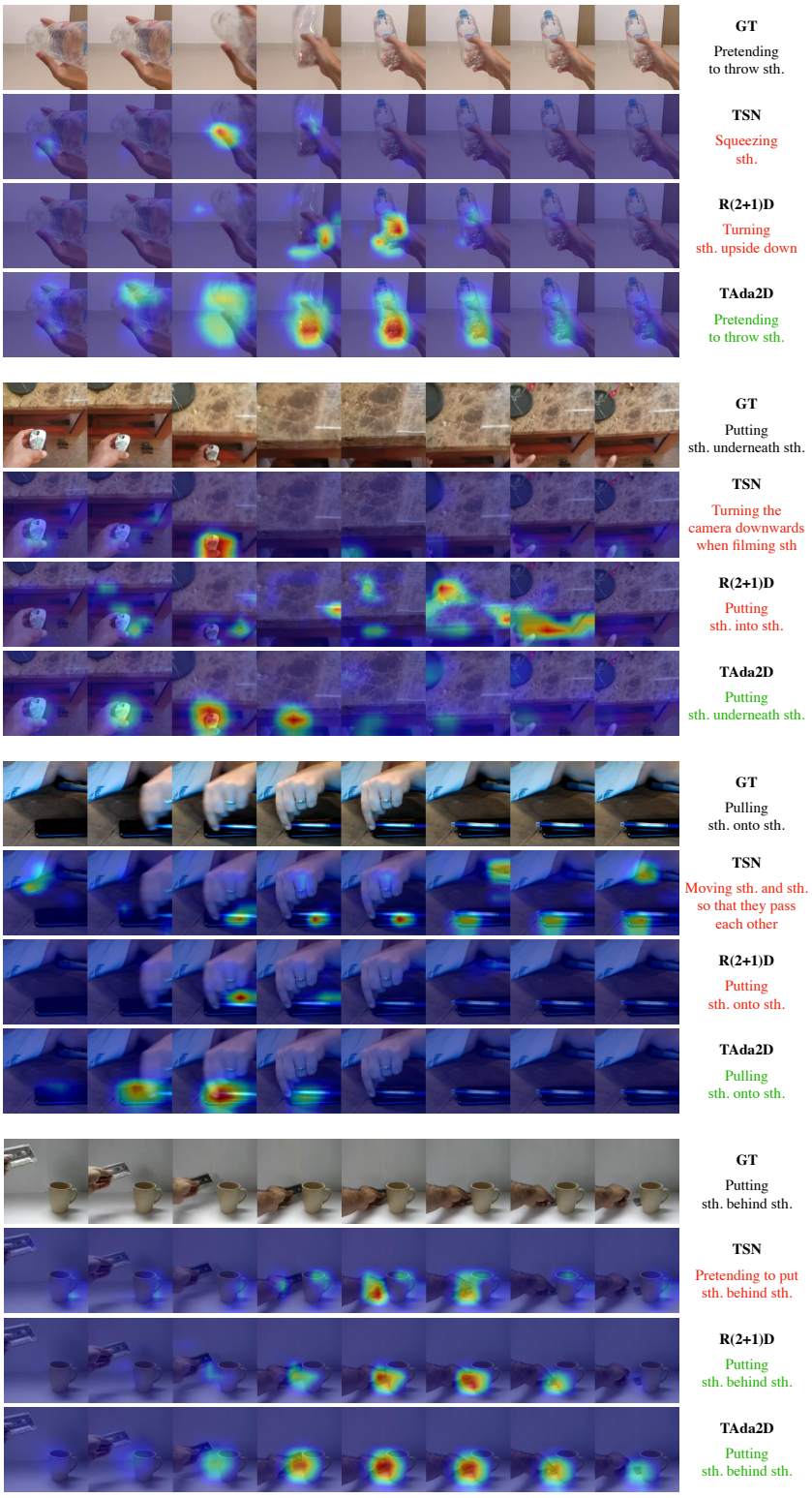


Figure A3: **Further qualitative evaluations on the Something-Something-V2 dataset.** In most cases, TAda2D captures meaningful areas in the videos for the correct classification. Further, the activated region of TAda2D also lasts longer along the temporal dimension compared to other two models, thanks to the global temproal context in the weight generation function $\mathcal{G}$.