

TM-Adapter: Temporal Merge Adapter for Efficient Global Temporal Modeling

Woo Joo Hahm Seungwoo Jang* Hyeon Tak Kim* Daeun Lee* Kwangsu Kim[†]

Sungkyunkwan University, Republic of Korea

hahmwj@g.skku.edu {sewo, gusxkr1706, dlekdms7931, kim.kwangsu}@skku.edu

Abstract

We propose the Temporal Merge Adapter (TM-Adapter), a novel framework for image-to-video parameter-efficient transfer learning (PETL), specifically designed for temporal representation learning in video understanding. PETL has emerged as a practical strategy for adapting large-scale vision models to video tasks under limited computational budgets. However, existing PETL approaches suffer from local redundancy caused by highly similar consecutive frames, which limits the modeling of diverse temporal dependencies. To address this limitation, we introduce a lightweight merge-unmerge mechanism that temporally aggregates and redistributes token embeddings, enabling the model to capture diverse temporal patterns by mitigating redundancy. Furthermore, to effectively handle diverse temporal dependencies across different time scales, TM-Adapter introduces a single adapter module with two parallel branches, local and global adapters, each specialized in capturing complementary patterns at different temporal ranges. We validate our approach through experiments on Kinetics-400, Something-Something V2, and HMDB-51, demonstrating competitive performance compared to existing methods while maintaining high parameter efficiency.

1. Introduction

Image models have become standard visual backbones for video understanding tasks due to their strong spatial representation capabilities [1, 5, 41, 44, 47, 63]. However, applying these models to video tasks via full fine-tuning is computationally expensive, mainly due to the high dimensionality of video data and the growing size of modern vision architectures [2, 17–19, 31, 52, 55, 56]. To address this, partial fine-tuning strategies have been proposed, where lightweight temporal modules — such as RNNs or Transformers — are added on top of frozen image encoders

*These authors contributed equally as second authors.

[†]Corresponding author

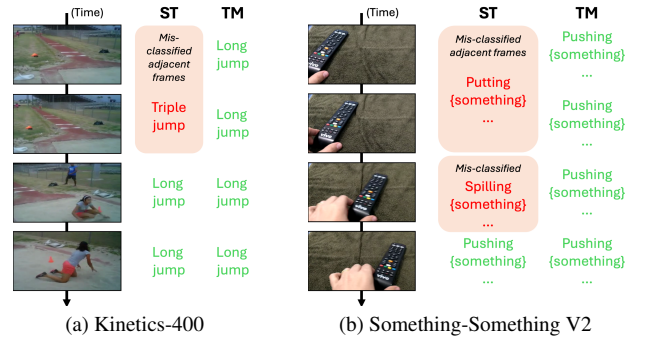


Figure 1. Comparison of frame-wise predictions between ST-Adapter (ST) [47], a representative image-to-video PETL method, and our method (TM-Adapter, denoted as TM). While ST-Adapter exhibits inconsistencies between individual frame predictions and the final video-level classification, TM-Adapter produces more consistent outputs across frames. Shown examples include (a) a “Long jump” action from the Kinetics-400 dataset and (b) “Pushing something from left to right” from the Something-Something V2 dataset [5, 20].

[19, 36, 38, 40, 65]. Although this reduces training cost, it often leads to suboptimal performance due to disjoint learning of spatial and temporal features. Recently, parameter-efficient transfer learning (PETL) has emerged as a more integrated approach, inserting lightweight temporal components directly into intermediate layers of frozen image models [27, 47, 50, 63, 64]. This design offers a better trade-off between performance and efficiency, making image-to-video adaptation more scalable and practical.

Although PETL methods have demonstrated strong performance on various video task, they often exhibit a bias toward local temporal information. For example, ST-Adapter [47] produces inconsistent framewise predictions even when trained on trimmed videos with a single label, suggesting dependence on short-term features rather than modeling global temporal context. Specifically, in the Kinetics-400 dataset [5], ST-Adapter misclassifies multiple frames in which the actor approaches the jump as “Triple jump”, despite the ground truth label being “Long

jump”, as shown in Figure 1a. This indicates that the model does not sufficiently capture long-range temporal dependencies, which are essential for distinguishing actions that are visually similar but semantically different. This limitation becomes more challenging in temporally complex datasets such as Something-Something V2 [20], where action classes are defined by fine-grained temporal dynamics rather than static visual features. As shown in Figure 1b, ST-Adapter generates inconsistent frame-level predictions within a single action instance, often jumping between semantically unrelated labels like *Putting...*, and *Spilling...* even though the ground truth is “*Pushing something from left to right*”. Such instability indicates that the model has difficulty capturing coherent temporal patterns, which are crucial for fine-grained temporal understanding.

To address these limitations, joint space–time attention has been proposed [2, 32, 33], but its computational complexity $O((T \cdot H \cdot W)^2 \cdot d)$ grows quadratically with the number of frames T , making it impractical for longer videos. Alternatively, two-stream architectures have been explored, in which separate streams are designed to operate on distinct patch embeddings or model branches in parallel [31, 48, 50, 64]. However, such designs introduce additional complexity in both model structure and parameter management. These challenges motivate two key questions: (1) Why do PETL methods tend to focus on local temporal information? (2) How can PETL models be designed to capture both local and global temporal dependencies efficiently?

In response to the first question, the bias toward local temporal information can be attributed to both the inherent characteristics of video data and the structural limitations of PETL methods. Firstly, video sequences are densely sampled and often exhibit minimal variation between adjacent frames, resulting in high temporal redundancy. This redundancy leads models to focus on short-term patterns, rather than modeling broader temporal structure needed to understand complex actions. From a modeling perspective, the limited trainable capacity of PETL methods introduced to maintain parameter efficiency not only restricts their ability to learn high-level temporal abstractions, but also causes them to inherit the inductive bias of pre-trained image backbones. Specifically, local temporal modeling is conceptually similar to tracking the main object across adjacent frames and follows a similar mechanism to how image models represent spatial relationships within a static image [4, 10]. Although recent PETL methods apply temporal modules in intermediate layers [24, 39, 47, 63], the dependence on a single architectural form of adapter restricts representational capacity, limiting the ability to overcome biases from redundancy.

To overcome these challenges, we propose the *Temporal Merge Adapter (TM-Adapter)*, an adapter module

composed of two parallel branches: a *Local-adapter* and a *Global-adapter*, each designed to capture complementary temporal dependencies at different scales. The *Local-adapter* focuses on capturing frame-level motion, while the *Global-adapter* targets long-range temporal dependencies by employing a merge–unmerge mechanism: it compresses the sequence by merging redundant frames based on inter-frame similarity, applies joint space–time attention to the compressed sequence, and restores the original temporal resolution to maintain compatibility with existing model pipelines. This architecture reduces the impact of temporal redundancy, enhances long-range temporal modeling by enabling efficient global reasoning over merged frames, and maintains parameter efficiency by utilizing only 14M additional trainable parameters on a frozen image backbone. Our main contributions are summarized as follows:

- We introduce the first trainable merge–unmerge mechanism for video understanding, which reduces spatio-temporal redundancy and enables efficient joint space–time attention.
- We are the first to introduce a dual-branch design within a single adapter module, internally splitting it into a *Local-adapter* and a *Global-adapter* to jointly learn short- and long-range temporal dependencies.

2. Related work

2.1. Image to Video Transfer Learning

Image models have served as effective backbone architectures for a variety of downstream video tasks, primarily due to their strong spatial representation capabilities. Early approaches extended 2D CNNs into the temporal domain by inflating them into 3D CNNs [5, 17, 19, 56] or by integrating temporal modules such as RNNs and LSTMs into higher layers of the network [36, 38, 65]. More recently, the emergence of Vision Transformers (ViTs) [14, 51], known for their superior generalization and transferability, has led to their widespread adoption in video understanding tasks [1, 2, 6, 15, 31–33, 37, 41, 44, 45, 55]. However, these advantages come at the cost of increased computational demands, as self-attention mechanisms require large-scale datasets and extensive training time.

2.2. PETL methods for Video

Adapter-based PETL methods were initially proposed in the NLP domain, where small trainable modules are inserted into frozen pre-trained language models to allow efficient transfer to downstream tasks [23, 49]. This idea has since been extended to vision tasks, particularly video understanding, by adapting spatial image models with lightweight temporal modules.

Existing image-to-video PETL methods are categorized into one-stream methods [6, 24, 26, 40, 47, 63] and two-

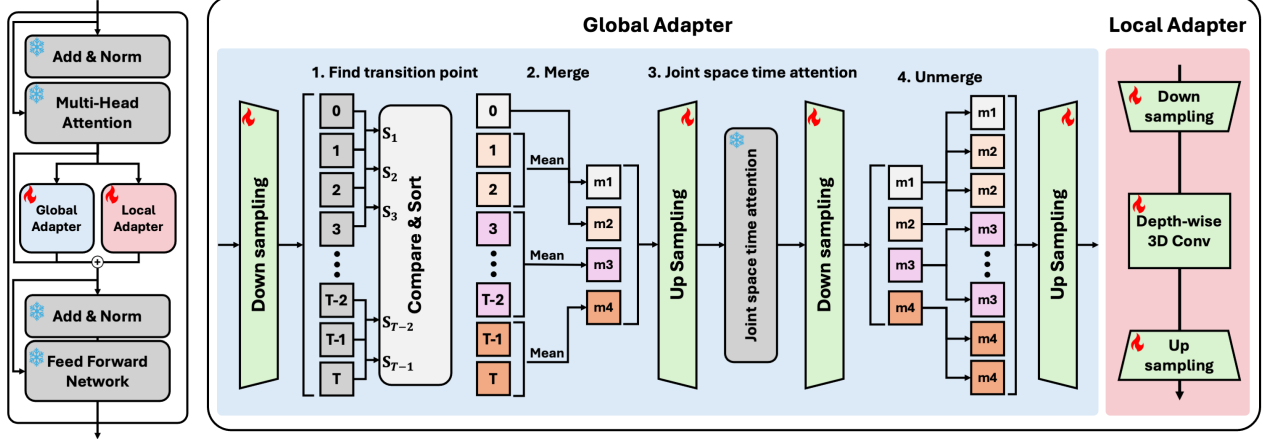


Figure 2. Overview of *TM-Adapter*. T , S , and m denote the number of frames, cosine similarity, and a merged frame, respectively (e.g., S_1 indicates the cosine similarity between frames 0 and 1, and m_0 refers to Merged Frame 0). In this figure, transition points are shown as $[S_0, S_2, S_{T-2}]$, which may vary across different videos. Icons indicate model components: the snowflake icon represents frozen modules, the fire icon indicates trainable components, and unmarked regions correspond to parameter-free operations.

stream methods [27, 39, 48, 50, 64]. One-stream methods augment frozen image models with lightweight temporal modules, allowing a single network to jointly model spatial and temporal representations. Representative one-stream methods include AIM [63], which introduces temporal-axis attention within transformer blocks, and ST-Adapter [47], which integrates depth-wise 3D convolution modules into intermediate layers. This approach simplifies the architecture and improves memory efficiency by training a single model. However, relying on a single type of temporal module with fewer trainable parameters often limits the capacity to capture complex temporal patterns, due to the constrained ability to jointly encode spatial and temporal representations within a single lightweight module.

In contrast, two-stream methods utilize separate networks to independently model spatial and temporal information [40, 48, 50, 64]. For example, the Dual-Path Adapter [48] employs two CLIP-based encoders with a simple bottleneck adapter to process different input modalities (2D spatial and 3D temporal). DIST [50] uses an R(2+1)D network as the temporal branch while distilling knowledge from a frozen CLIP encoder, and Side4Video [64] adds a spatial-temporal side network parallel to the main image encoder to reduce memory consumption while maintaining performance. However, two-stream methods require separate computational pipelines for each stream with distinct input modalities, resulting in inefficient memory utilization, higher computational overhead, and increased architectural complexity from managing separate training paths and explicit fusion strategies. Considering these trade-offs, we propose the *TM-Adapter*, a one-stream design that captures high-dimensional temporal information using only 2D spatial tokens, enabling more efficient and unified spatio-

temporal representation learning.

3. Method

3.1. Merge-Unmerge in Vision Tasks

Recent studies have explored merge mechanisms to reduce redundancy in vision tasks [3, 34]. A central issue lies in defining the merging criteria and execution strategy. Common solutions merge tokens with high cosine similarity via average pooling. While effective at reducing redundant representations, these operations are performed in a pairwise manner, inherently enforcing sequential computation and hindering efficient parallelization. Such serialization is tolerable at inference, where computational savings offset the overhead, but becomes a major drawback during training: merge-based algorithms are incompatible with gradient-based optimization, which demands parallelism and end-to-end differentiability.

In contrast, we propose the *TM-Adapter*, a learnable and parallelizable merge-unmerge framework specifically designed for global temporal modeling in image-to-video parameter-efficient transfer learning. Our method employs a learnable merge mechanism that supports training-time merging while preserving temporal order, without relying on handcrafted maps [34]. To enhance the diversity of learned representations, we further introduce a scale-aware attention mechanism that reflects the aggregation scale of merged tokens.

3.2. TM-Adapter

As shown in Figure 2, the *TM-Adapter* consists of two components: the *Local-adapter* and the *Global-adapter*. The *Local-adapter* processes the entire video input to capture

relationships between adjacent frames, while the *Global-adaptor* reconstructs the video into a shorter sequence to reduce local redundancy and emphasize global temporal information.

To improve computational and memory efficiency, we adopt a bottleneck structure that compresses the input tensor from $X \in \mathbb{R}^{D \times T \times W \times H}$ to $X \in \mathbb{R}^{(D/4) \times T \times W \times H}$. As shown in Figure 2, the *Down-Sampling* and *Up-Sampling* layers in both the local and global adapters implement this bottleneck structure.

3.3. Local Adapter

To capture short-term temporal dependencies across adjacent frames without affecting spatial resolution, we apply a lightweight depth-wise 3D convolution (*DW-3D-Conv*) with a kernel size of $(3 \times 1 \times 1)$. The input tensor $X \in \mathbb{R}^{DT \times (WH+1)}$ is reshaped to $X \in \mathbb{R}^{D \times T \times W \times H}$, excluding the class token, and the output is activated by GELU. This design efficiently encodes local motion while maintaining parameter efficiency, as formulated in Equation 1.

$$L(X) = \text{GELU}(DW\text{-}3D\text{-}Conv(X)) + X \quad (1)$$

3.4. Global Adapter

Unlike the *Local-adaptor*, which uses entire frames with redundant information to learn the relationships between adjacent frames, the *Global-adaptor* dynamically reconstructs the frames into a shortened version with reduced local redundancy. Specifically, to reconstruct the video, we compare the similarity between adjacent frames and merge similar frames within the *Global-adaptor*. To enable plug-and-play integration, the merged frames are restored to their original form through an unmerge step. However, parallelizing this process is challenging, as similar frames vary across videos. To address this, we divide the *Global-adaptor* into four steps—**Find transition points**, **Merge**, **Joint space-time attention**, and **Unmerge**—all of which are designed for parallel execution. This parallelized design improves scalability and achieves a 2.5× reduction in training time compared to a sequential implementation.

3.4.1. Find Transition Points

To identify frames exhibiting significant temporal shifts, we compute cosine similarity between consecutive frame embeddings, following Token Merging (ToMe) [3], which adopts cosine similarity as a redundancy metric due to its favorable balance between speed and accuracy. Specifically, we first apply average pooling along the channel dimension of the input tensor $X \in \mathbb{R}^{D \times (1+HW) \times T}$ to reduce computational cost and produce a compact frame-level representation, resulting in $X' \in \mathbb{R}^{(1+HW) \times T}$. We then define temporally adjacent frame pairs as $Prev = X'[:, 0:T-1]$ and $Aft = X'[:, 1:T]$, and compute their cosine similarity

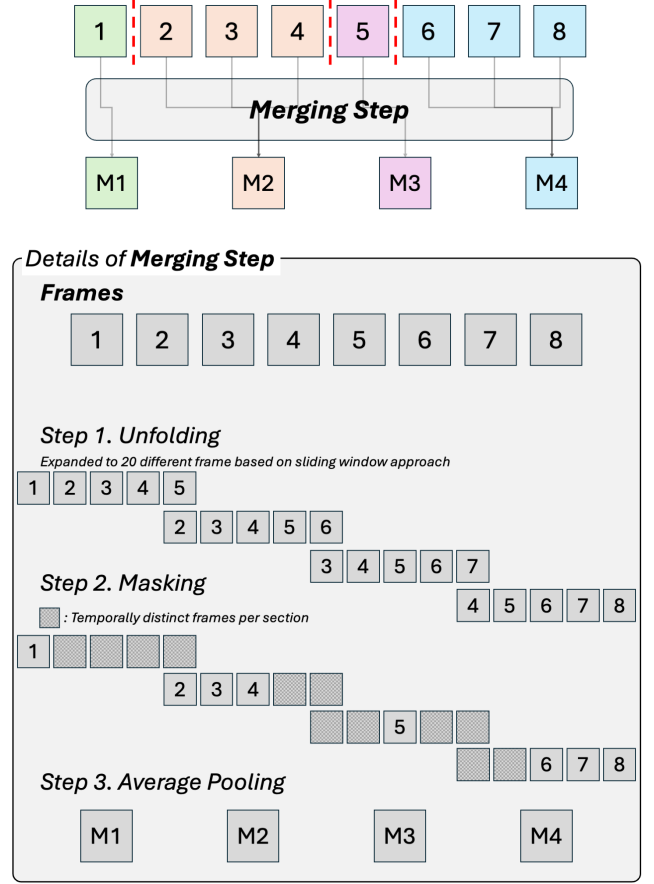


Figure 3. Overview of the Merging Step. This figure shows a parallelizable implementation of the merging process that aggregates redundant frames while mitigating the sequential constraint of temporal operations. The example assumes an 8-frame video with $MT = 4$ and transition points (e.g., $[0, 3, 4]$), which are identified in the Find Transition Points step and marked in red, while temporally distinct frames excluded from merging are shown in gray mask.

by L2-normalized inner product, where the main diagonal of $Prev^T Aft$ captures inter-frame similarity. The pairs of least similar frames K are selected as temporal transition points, where K is equal to the number of merge tokens (MT).

3.4.2. Merge

In the **Merge** step, frames are merged based on the transition points to shorten the video length from T to MT . To support parallel computation, this process is divided into three steps: **Unfolding**, **Masking**, and **Average pooling**.

Unfolding frames. To facilitate merging, frames are unfolded using a sliding window approach expanding the input from T frames to $MT \times kernel_size$, as illustrated in the *Unfolding* section of Figure 3. The *kernel_size*, defined as $(T - MT + 1)$, determines the maximum number of

Algorithm 1 Pseudo-code of define mask: Pytorch style

Input: $Idx \in \mathbf{R}^{1 \times MT}$ **Output:** weight

```
1:  $Idx = \text{torch.cat}([0], idx + 1, [T])$ 
2:  $\text{weights} = \text{torch.zeros}(MT, \text{kernel\_size})$ 
3:  $\text{stride} = \text{torch.arange}(MT)$ 
4:  $\text{scales} = Idx[:, 1:] - Idx[:, :-1]$ 
5:  $\text{end\_slice} = Idx[:, 1:] - \text{stride}$ 
6:  $\text{start\_slice} = \text{end\_slice} - \text{scales}$ 
7:  $\text{base} = \text{torch.arange}(\text{kernel\_size}) \times MT$ 
8:  $\text{mask} = (\text{base} \geq \text{start\_slice}) \& (\text{base} < \text{end\_slice})$ 
9:  $\text{weights}[\text{mask}] = 1$ 
10: return weights
```

frames that can be merged within a segment. For instance, when $T = 8$ and $MT = 4$, the kernel size is calculated as 5, which prevents imbalance when transition points are concentrated on one side (e.g., $[0, 1, 2]$) by ensuring that all remaining frames requiring merging (e.g., $[3, 4, 5, 6, 7]$) are properly grouped to maintain temporal consistency.

Masking. To identify frames for merging within each segment, a binary mask is generated based on the transition points (Idx). First, 0 and T are added at the beginning and end of Idx to define the full segmentation range. The merging *scales* are defined as the differences between consecutive indices in Idx , corresponding to the number of frames grouped between each pair of transition points. These *scales* are then used to compute the start and end slices in each unfolded window, and the mask is applied to select valid regions for merging, as described in the *Masking* step of Figure 3 and detailed in Algorithm 1.

Average pooling. To preserve temporal consistency and prevent scale distortion from uneven merging, each unfolded frame is normalized by the k/n_j , where k denotes the *kernel_size* and n_j is the number of merged frames in the j -th group. A 1D average pooling with *kernel_size* and stride ($T - MT + 1$) then reduces the sequence length from $MT \times k$ to MT .

$$\mathbf{A} = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}} + \log s \right) \quad (2)$$

3.4.3. Joint space time attention

After merging, we apply joint space-time attention to capture global temporal dependencies [2, 32, 33]. To allow the attention mechanism to reflect the temporal scope of each token and to prevent information loss caused by merging, we incorporate *scale* values into the attention logits, as described in Equation 2. These values, computed during the masking step in the **Merge** phase, represent the number of frames assigned to each token. In addition, we create a sin-

gle global class token that both captures global semantic information across merged segments and serves as the class token for the subsequent joint space-time attention. The attention is then computed using a frozen, pre-trained multi-head self-attention block to maintain parameter efficiency within PETL settings.

3.4.4. Unmerge

To reflect the learned global temporal representations in the original sequence, we reconstruct the full-length sequence by replicating each merged token to match its original temporal extent, expanding MT tokens back to T frames. For parallelization, we unfold each merged token as in the merging process, resulting in a tensor of shape $MT \times T$. A mask based on the merge size is then applied, followed by 1D average pooling with kernel size and stride equal to MT . Detailed pseudo-code is provided in the appendix.

4. Experiment

4.1. Datasets

We evaluate *TM-Adapter* on three action recognition benchmarks: Something-Something V2 (SSV2) [20], Kinetics-400 (K400) [5], and HMDB-51 (HMDB) [29]. SSV2 consists of 220k videos across 174 classes, focusing on how humans act in a continuous flow throughout the video. K400 consists of 300k videos across 400 classes, encompassing a wide variety of human actions. HMDB consists of 7,000 videos across 51 classes, comprising human actions extracted from movies and YouTube videos. Due to these characteristics, capturing global temporal information is crucial in SSV2, whereas spatio-temporal information in specific segments is more critical in K400 and HMDB, where a diverse range of discrete actions is observed.

4.2. Implementation details

Model. We utilized the pre-trained *CLIP-B/16* [51] image encoder as the backbone and attached a classifier customized for each benchmark dataset.

Training. All experiments were conducted with 8 input frames. Dense sampling was applied to K400 and HMDB to capture fine-grained motion, while uniform sampling was used for SSV2 to ensure coverage of the entire sequence.

Inference. For evaluation, we employed a multi-view frame sampling protocol, which extracts frames from multiple temporal segments and spatial viewpoints to ensure diverse feature representation and improve robustness.

4.3. Main results

Table 1 presents results on **Something-Something v2**, where our model achieves comparable performance to full fine-tuning while using significantly fewer trainable parameters (Video-Swin-B: 88M *vs.* 14M) and requiring lower

Table 1. Comparison with the state-of-the-art methods on the Something-to-Something V2 [20] dataset. Note that Views = frames \times clips \times spatial. GFLOPs marked with * are our measurements; others are cited from the paper.

Method & Arch.	Views	GFLOPs	Trainable # Params	SSV2 # R@1	SSV2 # R@5
Full-Fine tuning					
TimeSformer-L [2]	$64 \times 1 \times 3$	7140	121 M	62.4	-
UniFormer-B/16 [32]	$16 \times 3 \times 1$	777	60 M	70.2	92.8
ViViT FE [1]	$16 \times 4 \times 3$	11892	311 M	65.4	89.8
Video-Swin-B [41]	$32 \times 1 \times 3$	963	88 M	69.6	92.7
VideoMAE [55]	$16 \times 2 \times 3$	2880	87 M	69.7	92.3
VIT-B/16 [14]	$8 \times 1 \times 3$	281*	86M	44.0	77.0
MViTv2-B [15]	$32 \times 1 \times 5$	675	51 M	72.1	93.4
UniFormerV2-B/16 [33]	$16 \times 3 \times 2$	600	163 M	69.5	92.3
Parameter Efficient transfer learning					
AdaptFormer w/ B/16 [6]	$8 \times 1 \times 3$	544	8 M	51.3	70.6
EVL w/ B/16 [39]	$8 \times 1 \times 3$	512	29 M	61.0	-
Pro-Tuning w/ B/16 [46]	$8 \times 1 \times 3$	538	9 M	50.8	69.9
ST-Adapter w/ B/16 [47]	$8 \times 3 \times 1$	325*	14 M	67.1	91.2
AIM w/ B/16 [63]	$8 \times 1 \times 3$	624	14 M	66.4	90.5
Zeroi2v w/ B/16 [35]	$8 \times 1 \times 3$	422	20 M	67.7	90.8
TM-Adapter w/ B/16 (Ours)	$8 \times 2 \times 3$	403	28 M	68.3	91.6

Table 2. Comparison with the state-of-the-art methods on the Kinetics-400 [5] dataset. Note that Views = frames \times clips \times spatial. GFLOPs marked with * are our measurements; others are cited from the paper.

Method & Arch.	Views	GFLOPs	Trainable # Params	K400 # R@1	K400 # R@5
Full-Fine tuning					
UniFormer-B	$16 \times 1 \times 4$	777	60 M	82.0	95.1
SlowFast+NL [19]	$16 \times 3 \times 10$	7020	60 M	79.8	93.9
TimeSformer-L	$96 \times 1 \times 3$	7140	121 M	80.7	94.7
Video-Swin-B	$32 \times 4 \times 3$	963	88 M	82.7	95.5
X-CLIP [45]	$8 \times 4 \times 3$	1160	85 M	81.1	94.7
MViTv2-B	$32 \times 1 \times 5$	7200	51 M	82.9	95.4
UniFormerV2-B	$8 \times 3 \times 4$	1600	160 M	84.4	96.3
Parameter Efficient transfer learning					
EVL w/ B/16	$8 \times 3 \times 1$	444	29 M	82.9	-
ST-Adapter w/ B/16	$8 \times 3 \times 1$	303*	7 M	82.0	95.7
AIM w/ B/16	$8 \times 3 \times 1$	606	11 M	83.9	96.3
Zeroi2v w/ B/16 [35]	$8 \times 3 \times 1$	422	20 M	83.0	95.8
TM-Adapter w/ B/16 (Ours)	$8 \times 3 \times 2$	353	14 M	83.2	95.9

GFLOPs (ViViT FE: 11892 *vs.* 403). In PETL models, improvements were observed in the following: AdaptFormer (+ 17%) [6], EVL (+ 7.3%) [39], ST-Adapter (+ 0.8%) [47] and AIM (+1.9%) [63]. These results indicate that our approach effectively captures the general temporal dependencies of datasets that exhibit complex temporal patterns.

In **Kinetics-400**, similar to the results on Something-Something v2, our model achieves comparable performance to that of previous supervised video models, as shown in Table 2, while requiring significantly fewer trainable pa-

rameters (TimesFormer-L: 121M *vs.* 20M; UniFormerV2-B: 160M *vs.* 20M) and GFLOPs (UniFormerV2: 1300 *vs.* 353; X-CLIP: 1160 *vs.* 353). Furthermore, compared to PETL models, our approach demonstrated higher performance (EVL + 0.3%, ST-Adapter + 0.2%) and lower GFLOPs (EVL: 444 *vs.* 353; AIM: 606 *vs.* 353). While AIM achieved slightly higher overall accuracy (by 0.7%), we observe that our model performs better on SSV2, where capturing the global temporal flow is essential. This difference may stem from the intrinsic nature of Kinetics-400, in

Table 3. Comparison with the state-of-the-art methods on the HMDB dataset [29].

Method	Pretrain	Trainable # Params	HMDB # R@1
<i>Full-Fine tuning</i>			
VIT-B/16	CLIP	85M	59.4
I3D	K400	12M	74.8
R(2+1)D	K400	63M	74.5
Slowonly-R101	K400	60M	79.0
<i>PETL</i>			
VideoPrompt	CLIP	92M	66.4
ST-Adapter	K400	7M	77.7
DUALPATH	CLIP	10M	75.6
Ours	CLIP	14M	83.5

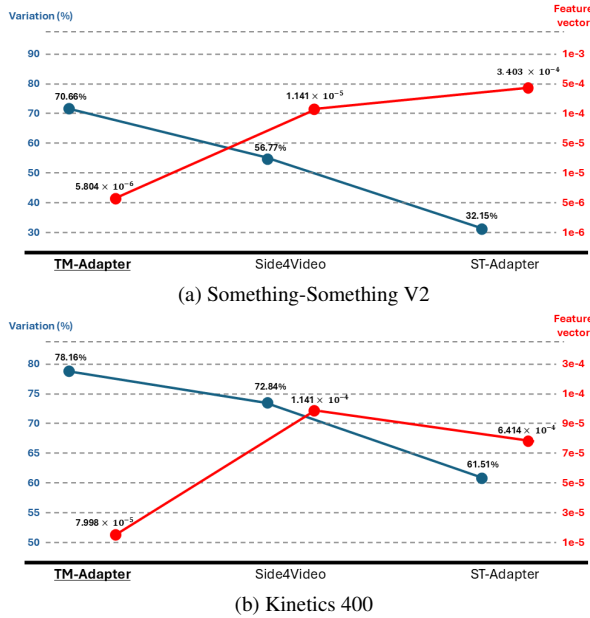


Figure 4. Comparison of frame consistency with and without the *Global-adapter*.

which spatial information from individual frames tends to be more critical than dynamic temporal context.

In **HMDB-51** (Table 3), *TM-Adapter* outperforms full fine-tuning models (I3D +8.7%, R(2+1)D +9.0%, Slowonly +4.5%) and PETL baselines (VideoPrompt +17.1%, ST-Adapter +5.8%, DUALPATH +7.9%). This indicates strong performance in short-range spatio-temporal scenarios and suggests that the model generalizes well even on smaller datasets.

4.4. Ablation studies

We perform comprehensive ablation studies to determine the most effective way to utilize *TM-Adapter*. For a fair ablation study, all experiments adopted uniform sampling

from the Something-Something V2 dataset, and all models were trained for 70 epochs using the same hyper-parameter settings.

Frame-wise inconsistency. To assess consistency between frame-level predictions and global video understanding, we adopt two metrics: Mean Absolute Difference (MAD) and Jensen-Shannon Divergence (JSD). As shown in Figure 4a and Figure 4b, *TM-Adapter* consistently achieves higher agreement rates and lower divergence than both one-stream and two-stream baselines, highlighting its ability to incorporate global video information into frame-level predictions. Concretely, *TM-Adapter* improves consistency on SSV2 by over 14%p compared to ST-Adapter, and also achieves a 16%p gain on K400, confirming substantial benefits in temporal coherence.

Efficiency of Merge-Unmerge and Parallelization. Table 4 compares the computational cost and training efficiency of different configurations of *TM-Adapter*. The base model, which uses joint space-time attention without the merge-unmerge mechanism, has a high computational load of 22.50B FLOPs. However, without parallelization, training becomes slower due to sequential operations and CPU bottlenecks, taking 1 hour 25 minutes per epoch on an A100 GPU. By incorporating parallelization techniques, the training time is reduced to 35 minutes per epoch while maintaining the same FLOPs. These results highlight the importance of both architectural optimization and implementation efficiency in achieving scalable training performance.

Transition Points in Attention Maps. Although videos are annotated with a single label, they often consist of multiple distinct phases. For instance, the action “Moving something closer to something” includes structured stages such as approaching, moving, and withdrawing. Similarly, in a “riding a bike” sequence, the motion may be interrupted by static background frames. As shown in Figures 5a and 5b, *TM-Adapter* successfully detects these temporal boundaries, enabling the model to concentrate on key motion changes and disregard redundant segments. These results demonstrate that our method improves temporal segmentation and enhances action understanding in videos with complex dynamics.

In addition, we provide a quantitative evaluation using the Breakfast dataset [30], which contains 1.7k cooking videos annotated with fine-grained action boundaries. We report Boundary F1 scores, where *TM-Adapter* achieves 0.7559, surpassing ST-Adapter (0.7357). This result confirms that our transition point detection not only improves recognition accuracy but also yields more precise temporal segmentation boundaries.

Attention map of each adapter. To analyze the effectiveness of the *Local-adapter* and *Global-adapter*, we visualized the attention maps of both adapters. As shown

Table 4. Comparison of FLOPs and training time for different configurations.

Setup Description	Merge-Unmerge	Parallelization	FLOPs	Training Time / Epoch (A100)
Baseline (No Merge-Unmerge)	✗	✗	22.50B	—
TM-Adapter (no parallelization)	✓	✗	14.05B	1 hr 25 min
TM-Adapter (with parallelization)	✓	✓	14.05B	35 min

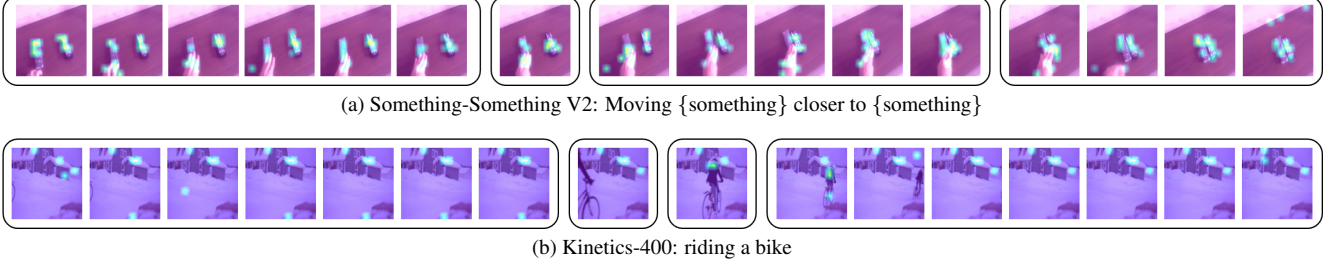


Figure 5. Visualization of attention maps to identify transition points across frames. Frames are merged based on transition points obtained with 16-frame input and a merge token value of 4.

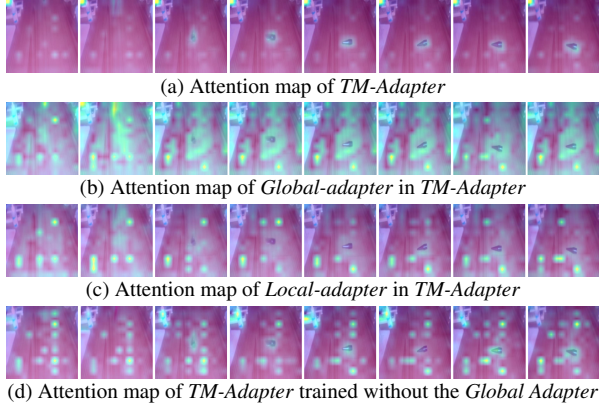


Figure 6. Visualization of attention maps to evaluate the effectiveness of *Local-adapter* and *Global-adapter*. The dataset is Something-Something V2 with the label “{Something} falling like a rock.”

in Figure 6c, the *Local-adapter* leads to local redundancy in short-range spatiotemporal information while focusing on irrelevant regions. In contrast, as shown in Figure 6b, the *Global-adapter* effectively captures changes around the main object, although it lacks detailed focus. The *TM-Adapter*, as shown in Figure 6a, combines the strengths of both and maintains consistent attention on the movement of the main object. To further examine the impact of the *Global-adapter*, a model consisting only of the *Local-adapter* was trained. The resulting attention map, as shown in Figure 6d, partially tracks the main object while still focusing on local redundancy and irrelevant regions.

5. Conclusion

This work presents *TM-Adapter*, a parameter-efficient transfer learning method for video action recognition. The proposed approach addresses two key challenges: the limited modeling capacity of PETL frameworks and the redundancy in consecutive video frames, which together hinder effective learning of global temporal representations. To mitigate these issues, we introduce a merge-unmerge algorithm that reduces temporal redundancy by identifying and aggregating similar frames during training while preserving motion dynamics critical for recognition. Furthermore, we design a parallel implementation of the merging process, reducing the training time by 2.5× under a fixed computational budget. *TM-Adapter* demonstrates strong performance in Kinetics-400, Something-Something V2, and HMDB-51, with fewer trainable parameters and lower computational cost compared to existing PETL methods. In addition, qualitative results indicate improved identification of temporal transitions and enhanced learning of coherent temporal structures.

Acknowledgements

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2024-00459618, Research on improving intelligent command and control capabilities based on generative AI and real-time 3D digital twin construction) and Next-Generation Intelligent Patrol Platform(Mobile police station) Program through the Korea Institutes of Police Technology(KIPoT) funded by the Korean National Police Agency. (No. RS-2025-25393280)

References

- [1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846, 2021. 1, 2, 6
- [2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, page 4, 2021. 1, 2, 5, 6
- [3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *The Eleventh International Conference on Learning Representations*, 2023. 3, 4
- [4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 2
- [5] 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*, pages 6299–6308, 2017. 1, 2, 5, 6
- [6] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678, 2022. 2, 6
- [7] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Multi-fiber networks for video recognition. In *Proceedings of the european conference on computer vision (ECCV)*, pages 352–367, 2018.
- [8] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. L2 regularization for learning kernels. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 109–116, 2009.
- [9] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR 2020 Workshops*, pages 702–703, 2020. 12
- [10] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations*, 2024. 2
- [11] 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*, pages 248–255. Ieee, 2009.
- [12] Ali Diba, Vivek Sharma, and Luc Van Gool. Deep temporal linear encoding networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2329–2338, 2017.
- [13] Ali Diba, Mohsen Fayyaz, Vivek Sharma, M Mahdi Arzani, Rahman Yousefzadeh, Juergen Gall, and Luc Van Gool. Spatio-temporal channel correlation networks for action classification. In *Proceedings of the European conference on computer vision (ECCV)*, pages 284–299, 2018.
- [14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. 2, 6
- [15] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6824–6835, 2021. 2, 6
- [16] Han Fang, Pengfei Xiong, Luhui Xu, and Yu Chen. Clip2video: Mastering video-text retrieval via image clip. *arXiv preprint arXiv:2106.11097*, 2021.
- [17] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 203–213, 2020. 1, 2
- [18] Christoph Feichtenhofer, Axel Pinz, and Andrew 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.
- [19] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019. 1, 2, 6
- [20] Raghu 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 international conference on computer vision*, pages 5842–5850, 2017. 1, 2, 5, 6
- [21] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 3154–3160, 2017.
- [22] 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*, pages 770–778, 2016.
- [23] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019. 2
- [24] Siteng Huang, Biao Gong, Yulin Pan, Jianwen Jiang, Yiliang Lv, Yuyuan Li, and Donglin Wang. Vop: Text-video cooperative prompt tuning for cross-modal retrieval. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6565–6574, 2023. 2
- [25] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

- [26] Shibo Jie, Zhi-Hong Deng, Shixuan Chen, and Zhijuan Jin. Convolutional bypasses are better vision transformer adapters. In *ECAI 2024*, pages 202–209. IOS Press, 2024. [2](#)
- [27] Xiaojie Jin, Bowen Zhang, Weibo Gong, Kai Xu, Xueqing Deng, Peng Wang, Zhao Zhang, Xiaohui Shen, and Jiashi Feng. Mv-adapter: Multimodal video transfer learning for video text retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27144–27153, 2024. [1](#), [3](#)
- [28] 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*, pages 1725–1732, 2014.
- [29] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International conference on computer vision*, pages 2556–2563. IEEE, 2011. [5](#), [7](#)
- [30] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 780–787, 2014. [7](#)
- [31] Dongho Lee, Jongseo Lee, and Jinwoo Choi. Cast: cross-attention in space and time for video action recognition. *Advances in Neural Information Processing Systems*, 36: 79399–79425, 2023. [1](#), [2](#)
- [32] Kunchang Li, Yali Wang, Peng Gao, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unified transformer for efficient spatial-temporal representation learning. In *International Conference on Learning Representations (ICLR)*, 2022. [2](#), [5](#), [6](#)
- [33] Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Limin Wang, and Yu Qiao. Uniformerv2: Spatiotemporal learning by arming image vits with video uniformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. [2](#), [5](#), [6](#)
- [34] Xirui Li, Chao Ma, Xiaokang Yang, and Ming-Hsuan Yang. Vidtope: Video token merging for zero-shot video editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7486–7495, 2024. [3](#)
- [35] Xinhao Li, Yuhao Zhu, and Limin Wang. Zeroi2v: Zero-cost adaptation of pre-trained transformers from image to video. In *European Conference on Computer Vision*, pages 425–443. Springer, 2024. [6](#)
- [36] 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*, pages 909–918, 2020. [1](#), [2](#)
- [37] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4804–4814, 2022. [2](#)
- [38] Ji Lin, Chuhan Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7083–7093, 2019. [1](#), [2](#)
- [39] Ziyi Lin, Shijie Geng, Renrui Zhang, Peng Gao, Gerard de Melo, Xiaogang Wang, Jifeng Dai, Yu Qiao, and Hongsheng Li. Frozen clip models are efficient video learners. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXV*, page 388–404, Berlin, Heidelberg, 2022. Springer-Verlag. [2](#), [3](#), [6](#)
- [40] Ruyang Liu, Jingjia Huang, Ge Li, Jiashi Feng, Xinglong Wu, and Thomas H Li. Revisiting temporal modeling for clip-based image-to-video knowledge transferring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6555–6564, 2023. [1](#), [2](#), [3](#)
- [41] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3202–3211, 2022. [1](#), [2](#), [6](#)
- [42] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.
- [43] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.
- [44] Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. Clip4clip: An empirical study of clip for end to end video clip retrieval. In *Proceedings of the 30th ACM international conference on multimedia*, pages 1823–1831, 2022. [1](#), [2](#)
- [45] Yiwei Ma, Guohai Xu, Xiaoshuai Sun, Ming Yan, Ji Zhang, and Rongrong Ji. X-clip: End-to-end multi-grained contrastive learning for video-text retrieval. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 638–647, 2022. [2](#), [6](#)
- [46] Xing Nie, Bolin Ni, Jianlong Chang, Gaofeng Meng, Chunlei Huo, Shiming Xiang, and Qi Tian. Pro-tuning: Unified prompt tuning for vision tasks. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(6):4653–4667, 2023. [6](#)
- [47] Juntong Pan, Ziyi Lin, Xiatian Zhu, Jing Shao, and Hongsheng Li. St-adapter: Parameter-efficient image-to-video transfer learning. *Advances in Neural Information Processing Systems*, 35:26462–26477, 2022. [1](#), [2](#), [3](#), [6](#), [15](#)
- [48] Jungin Park, Jiyoung Lee, and Kwanghoon Sohn. Dual-path adaptation from image to video transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2203–2213, 2023. [2](#), [3](#)
- [49] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, 2021. [2](#)
- [50] Zhiwu Qing, Shiwei Zhang, Ziyuan Huang, Yingya Zhang, Changxin Gao, Deli Zhao, and Nong Sang. Disentangling

- spatial and temporal learning for efficient image-to-video transfer learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13934–13944, 2023. [1](#), [2](#), [3](#)
- [51] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. [2](#), [5](#)
- [52] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, 27, 2014. [1](#)
- [53] Khurram Soomro, Amir Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, 2012.
- [54] Shixiang Tang, Dapeng Chen, Jinguo Zhu, Shijie Yu, and Wanli Ouyang. Layerwise optimization by gradient decomposition for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9634–9643, 2021.
- [55] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022. [1](#), [2](#), [6](#)
- [56] 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*, pages 4489–4497, 2015. [1](#), [2](#)
- [57] 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*, pages 6450–6459, 2018.
- [58] 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*, pages 5552–5561, 2019.
- [59] Amin Ullah, Jamil Ahmad, Khan Muhammad, Muhammad Sajjad, and Sung Wook Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE access*, 6:1155–1166, 2017.
- [60] 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*, pages 20–36. Springer, 2016.
- [61] 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*, pages 7794–7803, 2018.
- [62] 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)*, pages 305–321, 2018.
- [63] Taojiannan Yang, Yi Zhu, Yusheng Xie, Aston Zhang, Chen Chen, and Mu Li. Aim: Adapting image models for efficient video understanding. In *International Conference on Learning Representations*, 2023. [1](#), [2](#), [3](#), [6](#)
- [64] Huanjin Yao, Wenhao Wu, and Zhiheng Li. Side4video: Spatial-temporal side network for memory-efficient image-to-video transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages xxxx–xxxx, 2024. [1](#), [2](#), [3](#)
- [65] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George 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. [1](#), [2](#)
- [66] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018. [12](#)
- [67] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13001–13008, 2020. [12](#)

A. Appendix

In this document, we provide supplementary materials for “TM-Adapter: Temporal Merge Adapter for Efficient Global Temporal Modeling.” First, we present the implementation details, including training configurations and sampling strategies (Sec.B). Next, we outline the merge and unmerge procedures employed for efficient temporal processing (Sec.C). Finally, we provide attention map visualizations comparing TM-Adapter and ST-Adapter, illustrating the effectiveness of our approach in capturing temporal dynamics (Sec.D).

B. Implementation Details

As shown in Table 5, we present the implementation details for three benchmarks: Kinetics-400 (K400), Something-Something V2 (SSv2), and HMDB-51 (HMDB). The adapter bottleneck width is set to 192, providing twice the memory efficiency compared to a standard linear adapter design. For all datasets, the Merged Token (MT) count is set to 4, corresponding to the 8-frame input used during training. *TM-Adapter* is applied to layers 6–11 for K400 and HMDB, and to layers 0–11 for SSv2, reflecting the higher temporal variability in SSv2. Dense sampling is employed for K400 and HMDB, while uniform sampling is used for SSv2, with the sampling strategy chosen based on dataset characteristics. RandAugment [9] is applied across all datasets, while Random Erase [67] and MixUp [66] are specifically applied to SSv2 and HMDB to enhance robustness.

C. Detailed code of the merge and unmerge

C.1. Overview of the Merge Function

The *merge* function processes temporal information within the input tensor x by employing a predefined number of *merge_token*. The primary objective of this function is to efficiently condense temporal data while preserving essential features and minimizing redundancy across sequential frames. It operates on an input tensor of dimensions $X \in \mathbf{R}^{(B \times T) \times S \times P \times C}$, where \mathbf{B} represents the batch size, \mathbf{T} denotes the temporal length, \mathbf{S} corresponds to the sequence length, \mathbf{P} represents the spatial dimensions, and \mathbf{C} signifies the number of channels.

C.2. Find Transition Point

To detect meaningful transition points within the temporal sequence, the function first computes the mean of tensor x along the last dimension, optimizing computational efficiency. After performing a permutation, the last dimension is assigned as T (Temporal). The frames, excluding the final frame, are designated as *prev*, while those excluding only the first frame are defined as *after*. Each is normalized sep-

arately, followed by matrix multiplication, where the main diagonal of the resulting matrix represents the cosine similarity between consecutive frames. The *topK* operation is then applied to select the indices of the lowest *merge_token* - 1 similarity values, identifying them as transition points. The remaining consecutive frames are assumed to be similar, simplifying further computation. To align similarity indices with frame indices, 1 is added to each index, and sorting is performed to ensure efficient parallel processing in subsequent operations.

C.3. Define Mask

To define the frames to be merged within the temporal sequence, the function first appends 0 and T (the total sequence length) to the front and back of the transition point indices, forming the *base* and *last* tensors that represent the video boundaries. Then we calculate the number of frames to be merged within each section by computing the difference between adjacent indices, storing these values in the *indices* tensor. The *kernel_size*, which defines the maximum number of frames that can be merged in a single step, is determined as $(T - \text{merge_token} + 1)$ and is used during the unfolding operation. To construct a mask for selecting frames to be merged, a *weights* tensor initialized with zeros is prepared. The function generates *end_indices* by subtracting an arange tensor of size *merge_token* from the computed indices, while further subtraction produces *start_indices*. These indices define the frame selection boundaries within each sliding window. A logical operation is then applied on a *base_index* tensor representing the sliding window positions, setting values to 1 within the determined start and end boundaries in the *weights* tensor. This process effectively constructs a mask that identifies the frames to be merged in each sliding window, as illustrated in Figure 7.

C.4. Define Scale

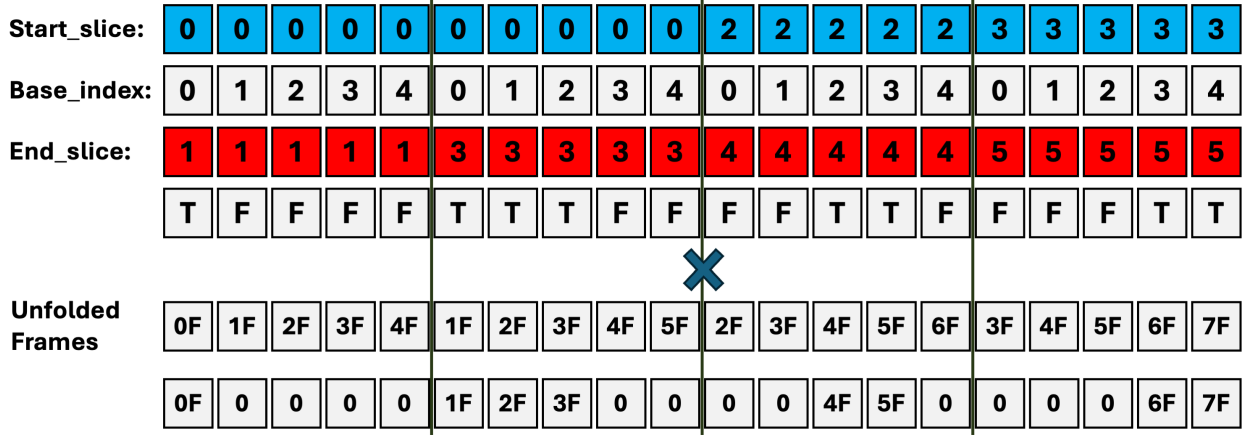
The *merge* function then calculates a scaling factor for each merging window to ensure that the merged values are normalized correctly. When the merged frames pass through the Joint Space-Time Attention layer, a single frame used in the computation consists of multiple frames. This means that the computation does not inherently account for the number of frames involved in the operation. To minimize the resulting information loss, the function defines a *scales* variable, which represents the number of merged frames in each segment. This scale is later used by applying the $\log(\text{scale})$ operation and adding it to the subsequent computations. The computed scales are then returned to ensure accurate representation.

C.5. Merge

Finally, the function performs the actual merging operation. The *unfold* method is applied to x_{id} to create slid-

Table 5. Detailed configuration settings across different benchmarks.

Components	K400	SSv2	HMDB51
Local-Adapter (MLP Mixer) per block	1	2	1
Global-Adapter (MLP Mixer) per block	1	1	1
Adapter (MLP Mixer) per block	3(1 Local, 2 Global)	4(2 Local, 2 Global)	3(1 Local, 2 Global)
Adapter bottleneck width	192	192	192
Merge Token	4	4	4
Layers	[6 : 11]	[0 : 11]	[6 : 11]
Optimizer			
AdamW, Cosine scheduler			
Learning rate	3×10^{-4}	5×10^{-4}	10^{-4}
Weight Decay	5×10^{-2}	5×10^{-2}	5×10^{-2}
Batch size	64	32	64
Layerwise Decay	0.75	0.75	0.75
Data configuration			
Training crop size	224	224	224
Frame sampling	8 for Dynamic Sampling	8 for Uniform Sampling	8 for Dynamic Sampling
RandAugment	✓	✓	✓
Random erase	X	✓	✓
MixUp	X	✓	✓
Inference configuration			
Testing views (temporal \times spatial)	3×5	2×3	2×3



Scaling + Average Pooling

Figure 7. Overview of the mask computation method and its application. The *T* and *F* under *End Slice* represent True and False, respectively, while *F* under *Unfolded Frames* denotes frames.

ing windows across the temporal dimension, enabling efficient parallel processing. The processed *weights* are then applied to these windows through element-wise multiplication, thereby merging the temporal information in a structured manner. Additionally, the *weights* are scaled by multiplying them with the ratio of *kernel.size/scales* to ensure that values are not inadvertently distorted during the subsequent average pooling operation. The merged data is

then processed through *Favg_pool1d*, which applies average pooling to further condense the temporal dimension. The resulting output is reshaped and permuted back to the format $B \times \text{merge_token} \times P \times C$, preparing it for the next stages in the model pipeline. The function concludes by returning the merged *output*, the calculated *scales*, and the cumulative *cum_indices*, completing the temporal merging process while preserving information integrity.

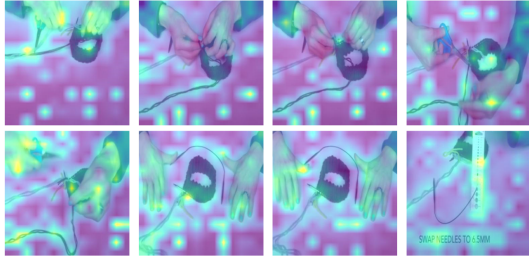
Algorithm 2 Pseudo-code of merge & unmerge

```
1: def merge( $x$ ,  $merge\_token$ ):
2:    $B, S, P, C = x.size()$ 
3:   # Find Transition points
4:    $x\_id = x.clone().permute(0, 2, 3, 1).contiguous().view(B, P \times C, T)$ 
5:    $x = torch.mean(x, dim = -1).permute(0, 2, 1)$ 
6:    $prev = F.normalize(x[:, :, -1], p = 2, dim = -2)$ 
7:    $after = F.normalize(x[:, :, 1:], p = 2, dim = -2)$ 
8:    $sim = torch.matmul(prev.transpose(-1, -2), after).diagonal(dim1 = -2, dim2 = -1)$ 
9:    $values, indices = torch.topk(sim, merge\_token - 1, dim = -1, largest = False, sorted = False)$ 
10:   $indices = torch.sort(indices, dim = -1)[0] + 1$ 
11:  # Define Mask
12:   $base = torch.zeros((B, 1), dtype = x.dtype, device = x.device)$ 
13:   $last = torch.ones((B, 1), dtype = x.dtype, device = x.device) \times T$ 
14:   $indices = torch.cat((base.expand(B, -1), indices, last.expand(B, -1)), dim = 1)$ 
15:   $cum\_indices = indices[:, 1:].clone()$ 
16:   $indices = indices[:, 1:] - indices[:, -1]$ 
17:   $kernel\_size = T - merge\_token + 1$ 
18:   $weights = torch.zeros(B, merge\_token \times (kernel\_size), dtype = x.dtype, device = x.device)$ 
19:   $weights = weights.view(B, merge\_token, kernel\_size)$ 
20:   $stride = torch.arange(merge\_token, dtype = indices.dtype, device = indices.device).expand\_as(indices)$ 
21:   $end\_slice = cum\_indices - stride$ 
22:   $start\_slice = end\_slice - indices$ 
23:   $start\_slice = start\_slice.unsqueeze(-1).expand(-1, -1, kernel\_size)$ 
24:   $end\_slice = end\_slice.unsqueeze(-1).expand(-1, -1, kernel\_size)$ 
25:   $base\_index = torch.stack([torch.arange(kernel\_size, device = x.device)] \times merge\_token)$ 
26:   $mask = (base\_index \geq start\_slice) \& (base\_index < end\_slice)$ 
27:   $weights[mask] = 1$ 
28:  # Define Scale
29:   $scales = mask.sum(dim = -1, keepdim = True, dtype = x.dtype).view(B, merge\_token)$ 
30:  # Merge
31:   $windows = x\_id.unfold(dimension = -1, size = kernel\_size, step = 1)$ 
32:   $weights = weights \times (kernel\_size / scales.unsqueeze(-1).expand(-1, -1, kernel\_size)).to(x.dtype)$ 
33:   $output = windows \times weights.unsqueeze(1).expand(-1, P \times C, -1, -1)$ 
34:   $output = output.contiguous().view(B, P \times C, merge\_token \times kernel\_size)$ 
35:   $output = F.avg\_pool1d(output, kernel\_size = kernel\_size, stride = kernel\_size)$ 
36:   $output = output.view(B, P, C, merge\_token).permute(0, 3, 1, 2)$ 
37:  return  $output, scales, cum\_indices$ 

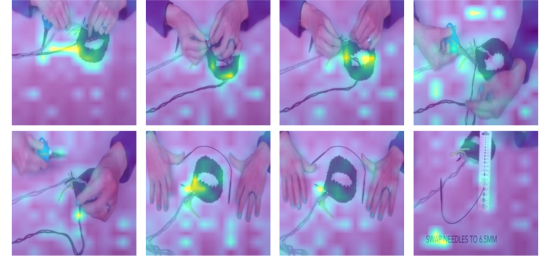
1: def unmerge( $x$ ,  $scale$ ,  $cum\_scale$ ,  $T$ ):
2:    $B, S, P, C = x.size()$ 
3:    $weights = torch.empty(B, S, T, dtype = x.dtype, device = x.device).fill_(0)$ 
4:    $start = torch.zeros((cum\_scale.size(0), 1), dtype = x.dtype, device = cum\_scale.device)$ 
5:    $cum\_scale = torch.cat((start, cum\_scale), dim = 1).unsqueeze(-1).expand(-1, -1, T)$ 
6:    $base\_index = torch.arange(T, dtype = x.dtype, device = x.device).expand\_as(weights)$ 
7:    $mask = (base\_index \geq cum\_scale[:, -1]) \& (base\_index < cum\_scale[:, 1:])$ 
8:    $weights[mask] = 1$ 
9:    $x = x.view(B, S, P \times C).unsqueeze(2).expand(-1, -1, T, -1)$ 
10:   $x = x \times weights.unsqueeze(-1)$ 
11:   $x = torch.sum(x, dim = 1).view(B, T, P, C)$ 
12:  return  $x$ 
```

D. Attention map Visualization

We conducted additional attention map visualizations in Figures 8 and Figure 9, comparing the ST-Adapter [47], which is similarly structured to the Local-Adapter, with the TM-Adapter. The experiments were performed on the Kinetics-400 and Something-Something V2 datasets. In Figure 8, focusing on the “Knitting” label, the ST-Adapter attention map concentrates on specific parts of the hands and redundant areas of the background Figure 8b. In contrast, the *TM-Adapter* provides a more balanced view of the entire image, directing attention not only to the object being knitted but also to the hands performing the knitting action 8a. Additionally, the *TM-Adapter* evenly reflects the entire frame in its results. As shown in Figure 9, a similar outcome was observed in the Something-Something V2 dataset. The label in Figure 9 is “Putting something, something, and something on the table,” which depicts a video of sequentially placing a cup, knife, and spoon on a table. While the ST-Adapter focuses only on the object currently being placed (Figure 9b), analyzing the relationships between adjacent frames, the *TM-Adapter* maintains attention on the objects previously placed as well (Figure 9a). This result indicates that the *TM-Adapter* effectively incorporates information from earlier frames into later ones, demonstrating its ability to focus not only on individual objects but also on the overall actions within the video.

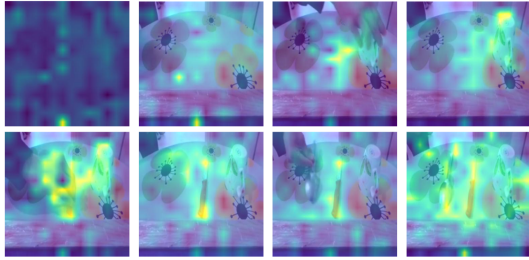


(a) TM-Adapter

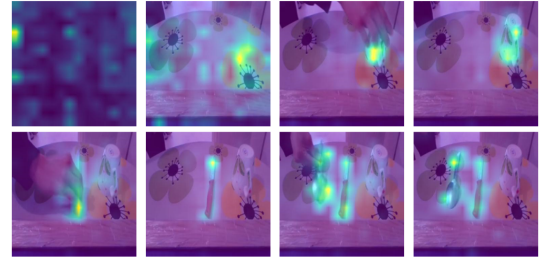


(b) ST-Adapter

Figure 8. Comparison of attention maps between the TM-Adapter and ST-Adapter for the “Knitting” label in the Kinetics-400 dataset.

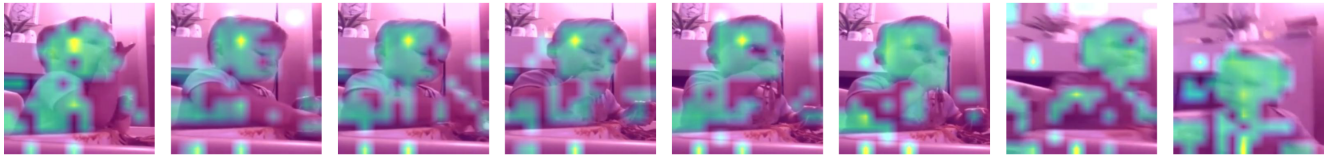


(a) TM-Adapter



(b) ST-Adapter

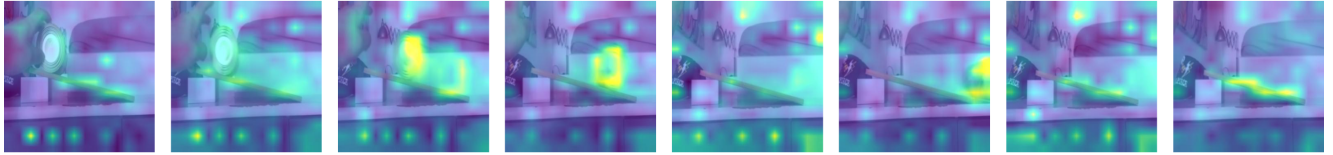
Figure 9. Comparison of attention maps between the TM-Adapter and ST-Adapter for the “Putting something, something and something on the table” label in the Something-Something V2 dataset.



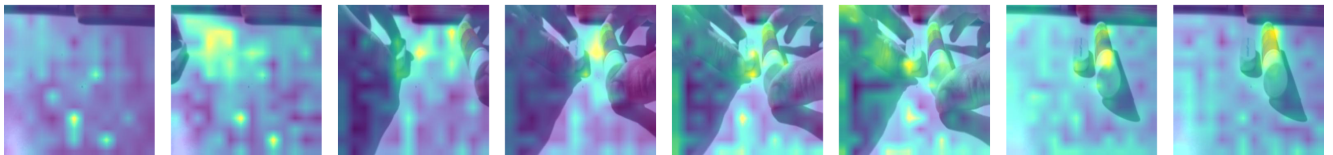
(a) Kinetics-400:Eating spaghetti



(b) Kinetics-400:Country line dancing



(c) Something-Something V2:Letting something roll down a slanted surface



(d) Something-Something V2:Moving something and something closer to each other

Figure 10. Additional attention maps of *TM-Adapter*.