FRAME ADAPTIVE NETWORK

Anonymous authors
Paper under double-blind review

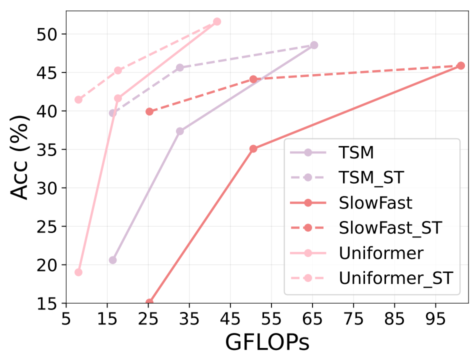
ABSTRACT

Existing video recognition algorithms always conduct different training pipelines for inputs with different frame numbers, which requires repetitive training operations and multiplying storage costs. If we evaluate the model using other frame numbers which are not used in training, our observation, named Temporal Deviation, shows the performance will drop significantly (see Fig. 1). Thus, the common training protocol for video related tasks is relatively rigid for flexible inference using various testing frames, especially for some edge devices with limited available frames or computational resources. In this study, we propose Frame Adaptive Network (FAN) to conduct a one-shot training but enable the model can be evaluated on different frame numbers. Concretely, FAN integrates several sets of training sequences, involves Specialized Normalization and Weight Alteration to efficiently expand the original network, and leverages Mutual Distillation for optimization. Comprehensive empirical validations using various architectures and popular benchmarks solidly demonstrate the effectiveness and generalization of FAN (e.g., 3.50/5.76/2.38% performance gain at frame 4/8/16 on Something-Something V1 dataset over competing method Uniformer), which also promises the practical potential of model usage.

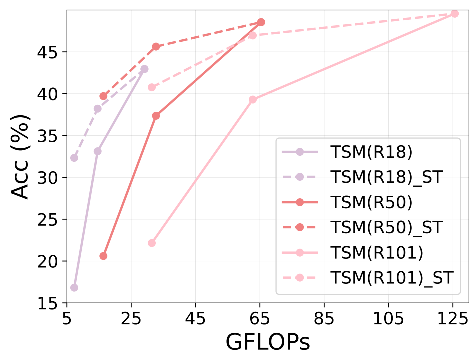
1 INTRODUCTION

The growing number of online videos boosts the research on video recognition, laying a solid foundation for deep learning which requires massive data. Compared with image classification, video recognition methods need a series of frames to represent the video which scales the computation. Thus, the efficiency of video recognition methods has always been an essential factor to evaluate these approaches. Specifically, the parameters and computations are two essential aspects to measure the efficacy of models which are related to storage size and inference speed, respectively. One existing direction to explore efficiency is designing lightweight networks Howard et al. (2017); Zhang et al. (2018) which are hardware friendly. Even if they increase the efficiency with an acceptable performance trade-off, these methods cannot make further customized adjustment to meet the dynamic-changing resource constraint in real scenarios. In community, there are two lines of research being proposed to resolve this issue. The first one is to design networks which can execute at various depths Huang et al. (2017) or widths Yu et al. (2018) to adjust the computations from the model perspective. The other line of research considers modifying the resolutions of input data Li et al. (2020a); Yang et al. (2020) to accommodate the cost from the data aspect. However, these methods are designed for image-based tasks, and building video recognition networks which can adjust the costs on-the-fly has been seldom touched.

Different from image related tasks, we need to pre-process the video data to avoid unaffordable costs which is a basic setting for both training and testing. A common practice for the pre-processing step is to: 1) sample a certain number of frames to represent the video; 2) feed them into various models, no matter whether the architecture is 2D Lin et al. (2019), 3D Feichtenhofer et al. (2019) or Transformer Li et al. (2022) model. Following this line, the computational costs will grow proportionally to the number of sampled frames. Concretely, standard protocol trains the same network with different frames separately to obtain multiple models with different performance and computations. This brings challenges to applying these networks on edge devices as the parameters will be multiplied if we store all models, and downloading and offloading models to switch them will cost non-negligible time. Moreover, the same video may be sampled at various frame rates on different platforms because of the difference in Internet speed, employing a single network which is trained at certain frame number for inference cannot meet the requirement in real scenarios.



(a) Temporal Deviation phenomenon exists in various video recognition architectures.



(b) Temporal Deviation phenomenon exists in different depths of deep networks.

Figure 1: Temporal Deviation phenomenon widely exists in video recognition. All methods are trained with high frame number and evaluated at other frames to compare with Separated Training (ST) which individually train the model at different frames on Something-Something V1 dataset.

Training the model with high frame number and directly evaluating it at fewer frames to adjust the cost is a naive and straightforward solution. To test its effectiveness, we compare it with training the model at different frames individually and testing with the corresponding frame numbers, named Separated Training (ST). We find there are significant performance gaps between the inference results and ST in all these architectures from Fig. 1 which means that these approaches should be evaluated at the same frame number used in training and will exhibit significantly inferior performance otherwise. Further, we conduct the same experiments on different depths of deep networks and the similar phenomenon proves that it is not related to the representation abilities of deep networks. We denote this generally existing phenomenon in video recognition as Temporal Deviation.

The potential reason for Temporal Deviation has been explored in Sec. 3 and briefly summarized as the shift in normalization statistics. To address this issue, we propose Frame Adaptive Network (FAN) which only needs to be trained for one-time, and can be adaptively evaluated at multiple frame numbers. We import several input sequences with different sampled frames to FAN during training and propose Specialized Normalization which gives different input sequences with private normalization operation. This is efficiency-friendly as normalization will introduce negligible parameters and computations. Besides, we introduce Weight Alteration to transform the shared weights of each sub-network to increase the representation capability and enable the network to exhibit strong performance at multiple frames. Further, we present Mutual Distillation which integrates Cross-Entropy and KL Divergence Loss to update the parameters of the whole network.

Compared with Separated Training (ST), FAN achieves higher accuracy at any frame with slightly extra parameters and computations. We validate FAN on various architectures and datasets, and FAN consistently outperforms ST which proves its effectiveness and generalization ability. With the proposed framework, we can resolve Temporal Deviation problem in video recognition and enable these methods to adjust their computations based on current resource budget by sampling different frames. Moreover, we provide a naive solution which enables FAN to be evaluated at any frame and increases its flexibility during inference. The validation results prove that FAN outperforms ST even at frames which are not used in training. We summarize the contributions as follows:

- We discover *Temporal Deviation* phenomenon: common training protocol for video recognition suffers from significant performance drop when testing frame number does not match with that of training. It is detailedly analyzed and practically inspires our study.
- We propose a general framework *Frame Adaptive Network* (FAN) to fix the problem brought by Temporal Deviation. It integrates several training sequences, involves Specialized Normalization and Weight Alteration to efficiently expand the original network, and leverages Mutual Distillation for optimization.
- Comprehensive empirical validations show that FAN, which only needs one-shot training, can adjust its computations by sampling different frames and outperform Separated Training (ST) at any frame on various architectures and datasets with negligible extra parameters and computations, which promises the potential application on edge devices.

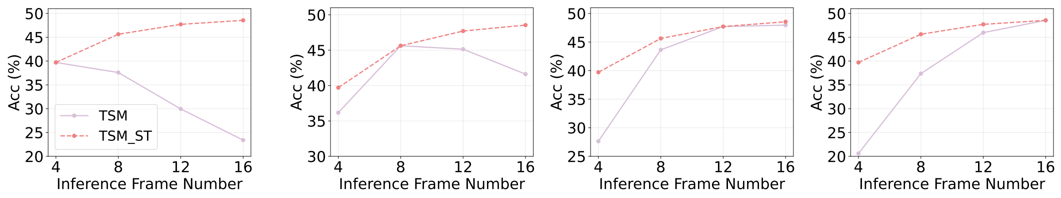
2 Related Work

Video recognition has been extensively explored in recent years and we can summarize the methods into three categories based on their architectures: 1) 2D-based models: TSN Wang et al. (2016) utilizes 2D networks as the backbone and averages the logits of sampled frames to represent the video prediction. Based on this framework, TSM Lin et al. (2019) is designed to shift part of the channels among adjacent frames to allow the exchange of temporal information. Recently, there are methods Li et al. (2020b); Wang et al. (2021) being proposed to take the global temporal information into account to achieve multi-scale temporal modeling. 2) 3D-based models: A straightforward solution for video recognition is to utilize 3D convolutions Tran et al. (2015); Carreira & Zisserman (2017); Feichtenhofer et al. (2019) which naturally consider the temporal information in frame sequences. Though effective, 3D convolutions bring the problem of huge computations. 3) Transformer-based models: Vision Transformers Dosovitskiy et al. (2020); Liu et al. (2021) have brought a research trend in computer vision due to its competitive performance. Based on ViT, many approaches Fan et al. (2021); Liu et al. (2022); Li et al. (2022) have been proposed recently for spatiotemporal learning and shown powerful performance which verifies the effectiveness of this structure to capture long-term dependencies.

Training-testing discrepancy widely exists in many scenarios of deep learning. FixRes Touvron et al. (2019) discovers the deviation of image resolutions between training and testing and found calibrating Batch Normalization (BN) Ioffe & Szegedy (2015) statistics plays an essential role at mitigating this phenomenon. Based on this observation, there are methods Li et al. (2020a); Yang et al. (2020) being designed to train a universal network to fit the images at different resolutions. Slimmable Neural Networks Yu et al. (2018); Yu & Huang (2019) train a shared network which can adjust its width to meet the resource constraints during inference. They design privatized BN for different switches because of the shift in BN statistics and introduce inplace distillation to further improve the performance of existing method. Different from these prior works, our work is motivated by Temporal Deviation phenomenon in video recognition. This finding is essential as frame sampling is a necessary step for all methods and former procedures train the network with different frames individually which is parameter-inefficient and memory-consuming.

Parameter-efficient transfer learning has aroused researchers' attention in NLP due to the arising of large-scale pre-trained language models. An important research line is to design task-specific adapters Pfeiffer et al. (2020a;b) which are inserted into the pre-trained model. In the fine-tuning stage, only the newly added adapter modules will be updated to achieve parameter-efficient. Recently, the idea of adapters has been extended to vision tasks as well and shown favorable performance Sung et al. (2022); Pan et al. (2022). In this work, instead of focusing on tuning from large-scale pre-trained models, we present Weight Alteration to transform the shared weights which increases the representation abilities of sub-networks.

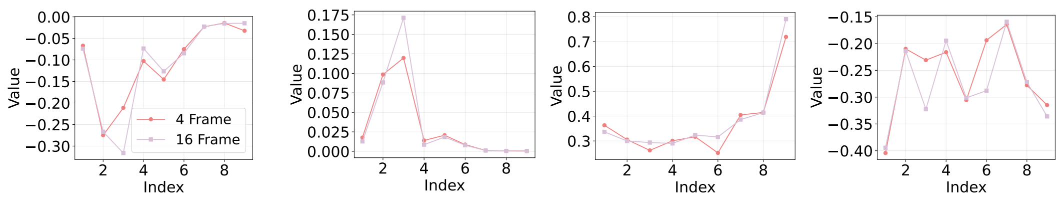
3 Temporal Deviation



(a) 4 Frame Training. (b) 8 Frame Training. (c) 12 Frame Training. (d) 16 Frame Training.

Figure 2: Nearby Alleviation phenomenon. TSM model is trained at different frame numbers in each figure on Something-Something V1 dataset and will be evaluated at all frames.

Nearby Alleviation. We can observe Temporal Deviation for various methods when they are trained with high frame numbers but evaluated at fewer frames from Fig. 1. To step further, we train TSM Lin et al. (2019) at other frame numbers and evaluate them at all frames. It can be seen from Fig. 2 that there are performance gaps for all models if it is not evaluated with the same frame number which is used in training. Particularly, the discrepancies vary in terms of the value and the performance gap is smaller if the inference frame is close to the training frame number. We denote this phenomenon as Nearby Alleviation because Temporal Deviation is less severe at nearby frames.



(a) Mean: μ (b) Variance: σ^2 (c) Scale: γ (d) Bias: β

Figure 3: Batch Normalization statistics of different models at various layers. Specifically, TSM models are trained at 4 frame and 16 frame separately, and the statistics are calculated from the fourth stage of ResNet-50.

Normalization Shifting. Batch Normalization (BN) Ioffe & Szegedy (2015) is proposed to reduce the internal covariance shift in deep networks and has been widely employed in convolutional neural networks due to its effectiveness. Specifically, it calculates the mean μ and variance σ^2 in a mini-batch for normalization and restores the representation power by scaling and shifting:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i, \quad \sigma^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x_i - \mu\right)^2, \quad y_i = \gamma\frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{1}$$

where γ and β are two learnable parameters and ϵ is a constant introduced to avoid zero value on denominator. In the standard training protocol, the model will be trained at a certain frame number and evaluated at the same temporal frequency. Temporal Deviation phenomenon will appear if the model is evaluated at other frame numbers. For example, if we train the model with v^H which has high frame number and evaluate it with low frame number v^L, the input of BN will be feature x^L and the corresponding output is:

$$y^{L'} = \gamma^H \frac{x^L - \mu^H}{\sqrt{{\sigma^H}^2 + \epsilon}} + \beta^H, \tag{2}$$

where μ^H, ${\sigma^H}^2$ are calculated at the data distribution of v^H, and γ^H, β^H are learnt at the training process with v^H. But the problem is whether the BN statistics of the models trained with different frames are the same? We calculate the statistics of the models trained with v^L and v^H separately and show it in Fig. 3. We can observe a discrepancy of BN statistics at different frame numbers. Note that μ and σ^2 are data-dependent which means that the divergence lies in data intrinsically. Thus, we conjecture that the discrepancy of BN statistics at different frames is an essential factor which leads to Temporal Deviation problem. Layer Normalization (LN) Ba et al. (2016) has been extensively used in Transformer-based models and its statistics are calculated in a similar way with BN which are related to the data distribution. Therefore, we believe the discrepancy of LN statistics is also one of the reasons for Temporal Deviation on Transformer-based models.

4 Frame Adaptive Network

In this section, we first present the training and inference paradigms of Frame Adaptive Network (FAN). Then, we propose Specialized Normalization to resolve the issue of Normalization Shifting. Further, we introduce Weight Alteration which is inserted into each sub-network to increase the representation ability. In the end, we elaborate Mutual Distillation for the optimization of FAN. Note that FAN is a general framework which can be built on different architectures (shown in Sec. 5.2) and we just take CNN based method as an example in this part for easier description.

4.1 Framework

The goal of our work is to present a method which can be evaluated at multiple frames and exhibit similar or even better performance compared to Separated Training (ST), which individually trains the model at different frames and evaluates them with the corresponding frames. Based on the analysis in Sec. 3, Temporal Deviation will appear if the model is evaluated at the frame which is not included during training. Therefore, we decide to import several sequences with different sampled frames to FAN shown in Fig. 4. Consider video v which is sampled at increasing frame numbers L, M and H, we can obtain v^L, v^M and v^H with temporal frequency of Low, Medium and

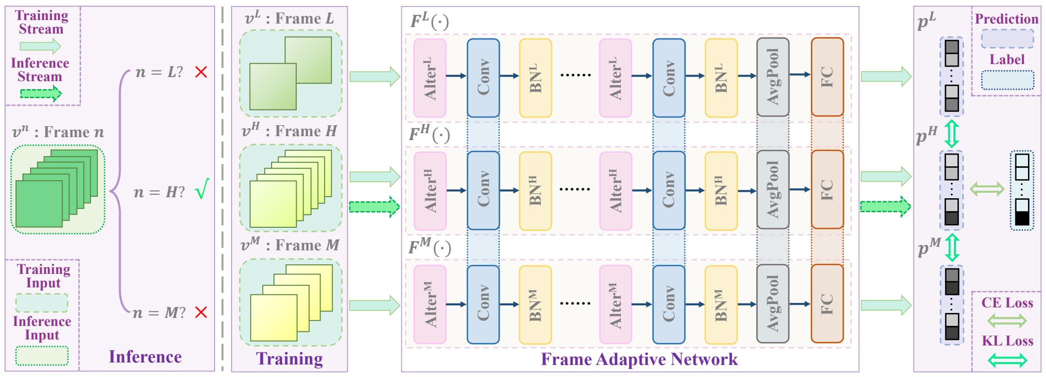


Figure 4: Illustration of Frame Adaptive Network (FAN). During training, given inputs with different frames v^L, v^M and v^H, we specialize normalization and add private weight alteration to construct three sub-networks $F^L(\cdot)$, $F^M(\cdot)$ and $F^H(\cdot)$ accordingly. We calculate CE loss on p^H and KL loss on p^L, p^M to update the parameters. During inference, we activate the sub-network which has the corresponding frame number with the input. We take CNN based model as an example.

High, respectively. These three sequences will be utilized at training phase to construct three sub-networks $F^L(\cdot)$, $F^M(\cdot)$ and $F^H(\cdot)$ accordingly. As for the inference paradigm, we will activate the sub-network which has the corresponding frame number with the input. In this manner, we can evaluate FAN with multiple frames during inference and adjust the computational costs accordingly.

4.2 Specialized Normalization

Based on our analysis in Sec. 3, Normalization Shifting is one of the reasons which leads to Temporal Deviation as the statistics μ, σ^2, γ and β are different for inputs with different sampled frames. Evaluating at other frames will use the statistics which are calculated at other distributions and lead to shifting in the feature space. Formally, we denote the intermediate features for v^L, v^M and v^H as x^L, x^M and x^H, respectively. We propose Specialized Normalization which provides v^L, v^M and v^H with private normalization:

$$y^* = \gamma^* \frac{x^* - \mu^*}{\sqrt{\sigma^{*2} + \epsilon}} + \beta^*, \tag{3}$$

where $* \in \{L, M, H\}$, and specialized normalization will learn its own γ and β and calculate the corresponding μ, σ^2 during training. In this way, the BN and LN can independently normalize the features of different frames during training and inference without interfering each other. Note that Specialized Normalization introduces negligible computations and parameters as normalization operation is a simple transformation and its parameters are often less than 1% of the model size.

In this manner, we construct three sub-networks with specialized normalization and the rest parameters are shared. But whether we can find a set of parameters for convolutions which is shared and display strong representation ability at different frames during inference remains a problem.

4.3 Weight Alteration

Ideally, specializing convolutions for different input sequences will enable the network to exhibit good performance as each sub-network learns the specialized parameters for the corresponding frame number which is similar with Separated Training (ST). Nevertheless, it will triple the parameters of the single network which makes it difficult for storage on edge devices. Sharing the weights of convolutions across $F^L(\cdot)$, $F^M(\cdot)$ and $F^H(\cdot)$ will make it parameter-efficient, but at the risk of limiting the representation capability of FAN.

Considering a shared convolution with weights W, the outputs of different sequences are:

$$y^* = W \otimes x^*, \tag{4}$$

where $* \in \{L, M, H\}$ and $\otimes$ stands for convolution which applies the same transformation for different input sequences. We propose Weight Alteration to transform the shared weight of each sub-network to diversify the weights and strengthen their representation abilities by:

$$y^* = \theta^* \otimes W \otimes x^*, \tag{5}$$

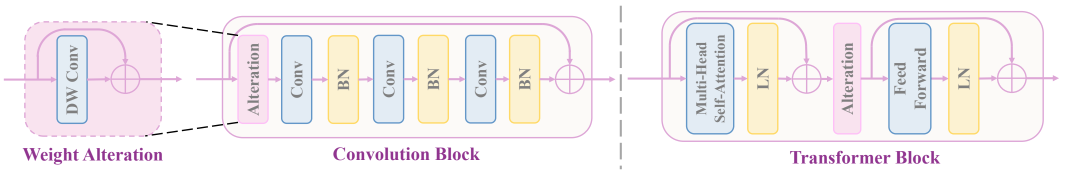


Figure 5: Specific designs of Weight Alteration, Convolution Block and Transformer Block. Weight Alteration is a Depth-wise convolution layer with a residual structure and we insert it into each Convolution Block and Transformer Block in Frame Adaptive Network (FAN).

which can also be written as:

$$y^* = W^* \otimes x^*, \quad W^* = \theta^* \otimes W, \tag{6}$$

where θ is a Depth-Wise convolution layer Chollet (2017) at each Convolution Block which can covert the shared weights W into specialized weights W^*. In this way, we can privatize the weights of convolutions for different sequences by a simple and efficient transformation. Given that video recognition methods often use pre-trained models, we include the residual structure He et al. (2016) to avoid the added module breaking the original computational graph of pre-trained models and restore their behaviors. Similarly, we also include the Weight Alteration in Transformer Block and we choose the inserted location following Zhang et al. (2022) shown in Fig. 5. Note that Depth-Wise convolution is lightweight and will introduce negligible parameters, the computation and parameters of FAN at inference stage will be similar with the original method.

4.4 Mutual Distillation

Given three input sequences v^L, v^M and v^H, we can obtain the predictions p^L, p^M and p^H accordingly. In most cases, video recognition models trained with v^H have better performance as the network will have access to more information of the original video. Therefore, we consider p^H is the most '*accurate*' prediction among the three as v^H has the most sampled frames. Applying Cross-Entropy loss on p^H, we can update $F^H(\cdot)$ by:

$$\mathcal{L}_{CE} = -\sum_{k=1}^{K} \hat{y}_k \log\left(p_k^H\right), \tag{7}$$

where $\hat{y}_k$ is the one-hot label of class k and there are K classes in total. Considering this loss independently, optimizing it will update the parameters of shared convolutions and specialized normalization, alteration in $F^H(\cdot)$. Directly calculating CE loss on p^L and p^M is a straightforward solution to update the parameters in $F^L(\cdot)$ and $F^M(\cdot)$, but it will lead to some problems. Firstly, the weights of convolutions are shared across three sub-networks and the optimal parameters for v^L after optimization may not fit well to v^M and v^H. Moreover, optimizing CE loss of p^L and p^M will lead to less favorable parameters of convolutions compared to only calculating Eq. 7 as their inputs contain less information compared to v^H which may lead to inferior performance.

Consequently, we utilize KL divergence Kullback (1997) loss to involve p^L and p^M in the computational graph and update the parameters of $F^L(\cdot)$ and $F^M(\cdot)$ using:

$$\mathcal{L}_{KL} = -\sum_{k=1}^{K} p_k^H \log\left(\frac{p_k^M}{p_k^H}\right) - \sum_{k=1}^{K} p_k^H \log\left(\frac{p_k^L}{p_k^H}\right). \tag{8}$$

As the weights of convolutions are shared across the three sub-networks, optimizing Eq. 8 will enforce the predictions of student (p^L and p^M) and teacher (p^H) networks to be as similar as possible and transfer the good knowledge from $F^H(\cdot)$ to $F^L(\cdot)$ and $F^M(\cdot)$. This can be achieved by only adjusting the parameters of specialized normalization and alteration as the weights of convolutions are shared. Considering the two losses in a uniform manner, we update the parameters of FAN by:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \cdot \mathcal{L}_{KL}, \tag{9}$$

where λ is an introduced hyperparameter to balance the two terms and we simply let $\lambda = 1$ in our implementations without fine-tuning the hyperparameter. With it as our optimization goal, we aim to find the parameters of convolutions which can be adapted into different input frames, and specialized parameters of normalization and alteration which are associated with corresponding input.

5 Empirical Validation

In this part, we validate Frame Adaptive Network (FAN) extensively on various architectures and benchmarks. First, we provide several baseline solutions for Temporal Deviation and compare them with FAN. Further, we apply our method on different methods and datasets to prove its generalization ability. Moreover, we provide a naive inference paradigm to enable FAN to be evaluated at any frame which promises its application on edge devices. Finally, we analyze the results of different numbers of input sequences and conduct ablations to analyze the effectiveness of our designs.

5.1 Experimental Preparation

Datasets. We conduct experiments on four datasets, including: (1) Something-Something V1&V2 Goyal et al. (2017) are two human action recognition datasets which includes 98k and 194k videos, respectively. They contain strong temporal dependency and show the most significant Temporal Deviation phenomenon among all datasets. (2) Kinetics400 Kay et al. (2017) is a large scale action recognition dataset with 400 action classes. (3) HMDB51 Kuehne et al. (2011) is composed of 6,766 videos which can be categorized into 51 classes. We adopt the original three training/testing splits for training and evaluation on this dataset.

Implementation details. We uniformly sample 4/8/16 frames for v_i^L, v_i^M and v_i^H for all methods except for SlowFast Feichtenhofer et al. (2019) which samples 16/32/64 frames for fast pathway. For the baseline results, we train all methods with v_i^H and evaluate them at v_i^L, v_i^M and v_i^H. Separated Training (ST) denotes training the network at v_i^L, v_i^M and v_i^H individually and evaluating them at the corresponding frame used in training. Other implementation details are included in appendix.

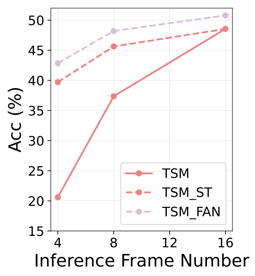
Baseline methods. As this topic has never been touched previously, it is difficult to compare FAN with other methods. In addition to Separated Training (ST) introduced before, we provide three more baseline methods for this problem: (1) **Mixed Sampling**: We sample 4 and 16 frames for v_i^L and v_i^H, respectively. Then we randomly choose 4 consecutive frames $v_i^{H'}$ from v_i^H and apply Mixup Zhang et al. (2017) between $v_i^{H'}$ and v_i^L to form the new data. The hyperparameter ρ decides the probability of whether to apply mixed sampling at each iteration. (2) **Proportional Sampling**: We let the network to randomly sample 4 frames or 16 frames at each iteration as this pair has the most significant Temporal Deviation phenomenon. The hyperparameter ρ denotes the probability to sample 16 frames for every iteration. (3) **Ensemble**: We take use of the models that are individually trained at 4,8 and 16 frames and averagely ensemble them to form a new model.

5.2 Main Results

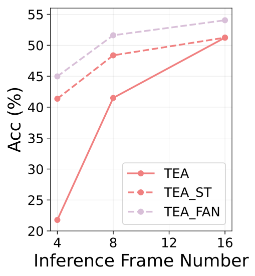
Table 1: Comparison with baseline methods on Something-Something V1 dataset. GFLOPs refers to the average computational cost to process a single video. The best results are bold-faced.

Method	Parameters	4 Frame		8 Frame		16 Frame	
		Acc.(%)	GFLOPs	Acc.(%)	GFLOPs	Acc.(%)	GFLOPs
TSM Lin et al. (2019)	25.6M	20.60	16.4	37.36	32.7	48.55	65.4
TSM-ST	25.6×3M	39.71	16.4	45.63	32.7	48.55	65.4
TSM-Mixed($\rho = 0.50$)	25.6M	27.89	16.4	41.07	32.7	48.44	65.4
TSM-Mixed($\rho = 0.75$)	25.6M	30.43	16.4	42.56	32.7	47.81	65.4
TSM-Proportional($\rho = 0.50$)	25.6M	37.56	16.4	44.82	32.7	45.37	65.4
TSM-Proportional($\rho = 0.75$)	25.6M	32.06	16.4	43.15	32.7	47.14	65.4
TSM-Ensemble	25.6×3M	35.88	16.4×3	46.25	32.7×3	46.82	65.4×3
TSM-FAN	25.7M	**42.85**	16.4	**48.20**	32.8	**50.79**	65.5

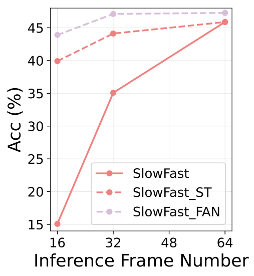
Comparison with baseline methods. Tab. 1 shows that the first two methods help to alleviate Temporal Deviation as the performance at frame 4/8 are better than the inference results of model trained with standard protocol. Nevertheless, the increase is obtained at the cost of a accuracy drop at high frame numbers. Then, we adjust the hyperparameter ρ and the results show that both methods seem to provide a trade-off solution for this problem: if the performance at low frames is better, the results at high frame numbers will be worse. Obviously, these two baseline methods do not provide satisfying answers for this problem as their performance is still worse than ST.



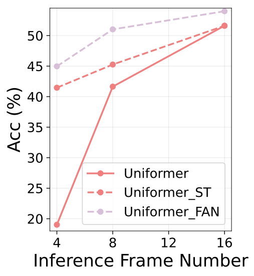
(a) Results on TSM.



(b) Results on TEA.

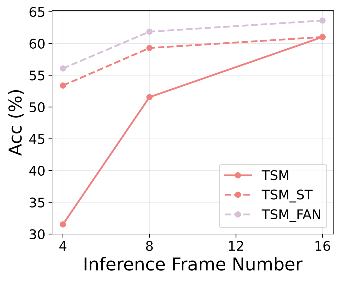


(c) Results on SlowFast.

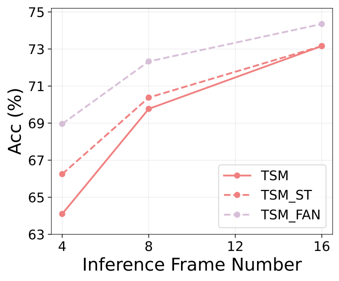


(d) Results on Uniformer.

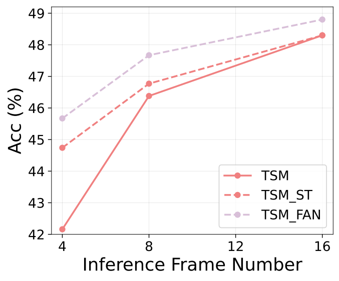
Figure 6: Validation results across different video recognition architectures on Something-Something V1 dataset, including 2D-network, 3D-network and Transformer-network.



(a) Results on Sth-Sth V2 dataset.



(b) Results on Kinetics400 dataset.



(c) Results on HMDB51 dataset.

Figure 7: Validation results across various video recognition datasets.

Ensemble outperforms ST at frame 8 at the cost of greater computational costs. Though, its performance at 4/16 frame is worse than ST which means that it still cannot effectively resolve Temporal Deviation problem. While our method FAN shows stronger results compared to ST and Ensemble at all frames with negligible added computations. Moreover, compared to ST and Ensemble which need to train the model separately for multiple times and storage the parameters for different models individually, our method is trained for only one time and introduces slightly extra parameters which makes it favorable for application on edge devices.

Performance analysis across architectures. We further validate our method on different architectures in Fig. 6. We first build our method on TSM Lin et al. (2019) which does not contain any parameters in the temporal modeling module. FAN exhibits advantages in performance at all frames compared to baseline TSM and ST. Then, we implement our method on TEA Li et al. (2020b) which involves convolutions and normalization in the temporal modeling module. Similarly, we specialize all normalization and share weight for all convolutions in TEA, and our results also surpass ST at all frames. Moreover, we extend FAN to other structures: SlowFast Feichtenhofer et al. (2019) and Uniformer Li et al. (2022). The results exhibit similar improvements at all frame numbers compared to ST. Particularly, our method enable competing method Uniformer to increase its accuracy at frame 4/8/16 by 3.50/5.76/2.38$\%$ which validates the effectiveness and generalization of our method.

Performance analysis across datasets. In this part, we empirically evaluate our method on various datasets in Fig. 7, including Something-Something V2, Kinetics400 and HMDB51. The first observation is that Temporal Deviation phenomenon is less obvious on Kinetics400 and HMDB51 as these two datasets contain less temporal information and the calculated normalization statistics of different input frames are similar. Nevertheless, FAN continuously improves the accuracy of ST on these datasets as well. For example, there are 2.71/1.95/1.19$\%$ performance gains at frame 4/8/16 on Kinetics400 which further demonstrate the validity of our design.

5.3 Inference at any frame

We have proved that FAN can outperform ST at the frame numbers used in training, but the evaluation at other frames which are not included in training remains untouched. Motivated by Nearby Alleviation in Sec. 3, we provide a naive inference paradigm to enable FAN to be evaluated at any frame. Given frame number n at inference phase, we will calculate the frame difference with L, M and H, and activate the sub-network with the minimal difference for validation. If the frame difference is the same for two sub-networks, we will choose the one which corresponds higher frame

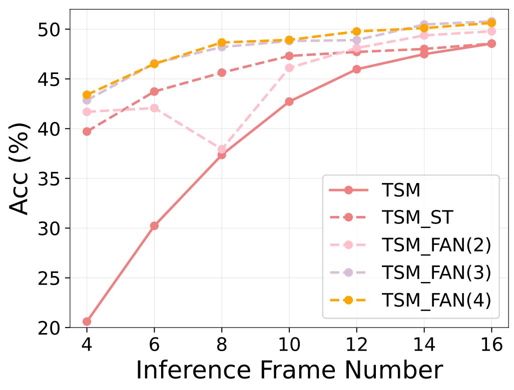


Figure 8: Validation results of FAN which imports different input sequences on Sth-Sth V1. We evaluate FAN at various frame numbers and compare with Separated Training (ST).

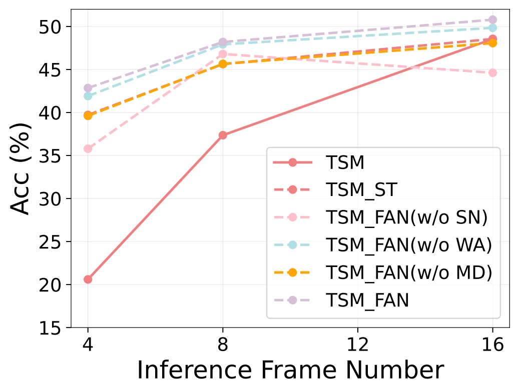


Figure 9: Ablation of design choices of FAN on Sth-Sth V1. SN denotes Specialized Normalization, WA stands for Weight Alteration and MD represents Mutual Distillation.

number by default. In this manner, we can evaluate other frame numbers which are not used in training and the results are included in Sec. 5.4.

5.4 Ablation

Input sequences combinations. As we input three frame sequences in our method, we can easily modify FAN by importing other numbers of input sequences. Shown in Fig. 8, we import two, three and four sequences to FAN respectively and evaluate their performance at various frames with the inference strategy in Sec. 5.3. To be specific, we import 4/16 frames to FAN(2), 4/8/16 frames to FAN(3) and 4/8/12/16 frames to FAN(4) in the training phase. First, we can observe that FAN(2) outperforms ST at 4/16 frames which are used in training, but its performance at 6/8/10 frames is worse than ST because of the missing middle sequence in training. In contrast, both FAN(3) and FAN(4) obtain higher accuracy at all frames compared to ST which can be attributed to the import of middle sequence in training so that the evaluation results at nearby frames can be guaranteed by Nearby Alleviation. FAN(4) achieves higher accuracy at frame 12 as it imports 12 frames during training. However, it will cost more time and resources during training with similar performance at other frames compared to FAN(3). Therefore, we adopt FAN(3) in all the experiments by default.

Design choices. In this section, we conduct ablation to verify the effectiveness of our designs in Fig. 9. First, we build FAN without specialized normalization so that the batch normalization are shared across the three sub-networks. We can observe obvious performance drop at 4/16 frames due to the shift in normalization statistics which proves the necessity of private normalization for different sub-networks. Further, we optimize FAN by calculating Cross-Entropy loss on the predictions of all sub-networks respectively and do not utilize KL divergence loss for optimization. The obtained curve is similar with ST and the worse results indicate that Mutual Distillation is a better solution as the combined CE loss will result in less favorable wights of the shared convolutions. Finally, we compare FAN with its variants which does not include Weight Alteration in the convolution block and FAN exhibits better performance at all frames. The reason is Weight Alteration transforms the shared weights of convolutions into private parameters for each sub-network and increases their representation ability at corresponding frame numbers.

6 Conclusion

In this paper, we discover Temporal Deviation phenomenon which widely exists in video recognition and propose Frame Adaptive Network (FAN) to address this issue. Specifically, we import several sequences with different frames into the network and design Specialized Normalization for every input. Further, we propose Weight Alteration to transform the weights and increase the representation ability of each sub-network. Finally, we present Mutual Distillation which combines CE and KL loss to update the parameters of the sub-networks. Extensive experiments demonstrate that FAN, which only requires one-shot training, can be evaluated at multiple frame numbers and outperforms Separated Training with negligible costs, making it favorable for applications on edge devices.

Ethics Statement

In our paper, we strictly follow the ICLR ethical research standards and laws. To the best of our knowledge, our method has no potential negative societal impacts and complies with the General Ethical Principles.

Reproducibility Statement

We adhere to ICLR reproducibility standards and ensure the reproducibility of our work. All datasets we employed are publicly available. Besides, we have included the training details in the appendix and we will provide the code to reviewers and area chairs using a link to an anonymous repository when the discussion forums is open.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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*, pp. 6824–6835, 2021.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 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 *ICCV*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

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.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

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.

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*, pp. 2556–2563. IEEE, 2011.

Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

Duo Li, Anbang Yao, and Qifeng Chen. Learning to learn parameterized classification networks for scalable input images. In *ECCV*, 2020a.

Kunchang Li, Yali Wang, Junhao Zhang, Peng Gao, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unifying convolution and self-attention for visual recognition. *arXiv preprint arXiv:2201.09450*, 2022.

Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *CVPR*, 2020b.

Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

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*, 2022.

Junting Pan, Ziyi Lin, Xiatian Zhu, Jing Shao, and Hongsheng Li. St-adapter: Parameter-efficient image-to-video transfer learning for action recognition. *arXiv preprint arXiv:2206.13559*, 2022.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020a.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*, 2020b.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *CVPR*, 2022.

Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. *Advances in neural information processing systems*, 32, 2019.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

Limin Wang, Zhan Tong, Bin Ji, and Gangshan Wu. Tdn: Temporal difference networks for efficient action recognition. In *CVPR*, 2021.

Taojiannan Yang, Sijie Zhu, Chen Chen, Shen Yan, Mi Zhang, and Andrew Willis. Mutualnet: Adaptive convnet via mutual learning from network width and resolution. In *ECCV*, 2020.

Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *ICCV*, 2019.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Jinnian Zhang, Houwen Peng, Kan Wu, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Minivit: Compressing vision transformers with weight multiplexing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12145–12154, 2022.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.

A APPENDIX

A.1 IMPLEMENTATION DETAILS

The training data is random cropped to 224 $\times$ 224 and we perform random flipping except for Something-Something datasets. At inference stage, all frames will be center-cropped to 224 $\times$ 224 except SlowFast Feichtenhofer et al. (2019) which adopts the resolution of 256 $\times$ 256 for evaluation. We use one clip per video for efficiency. We train all models on NVIDIA Tesla V100 GPUs and adopt the same training hyperparameters with the official implementations.

A.2 MORE SAMPLED FRAMES

Table 2: Experiments with more sampled frames on Something-Something V1.

Method	Top-1 Acc.(%)			
	4 Frame	8 Frame	16 Frame	24 Frame
TSM Lin et al. (2019)	13.91	27.91	43.59	47.90
TSM-ST	39.71	45.63	48.55	47.90
TSM-FAN	**41.28**	**46.72**	**49.79**	**49.95**

In previous experiments, we sample 16 frames at most to represent v^H. Therefore, we include more frames in this section and import 4 sequences with 4/8/16/24 sampled frames, respectively. The first observation from Table 2 is that the performance of TSM(24F) is even a little bit lower than TSM(16F) which can be attributed to simple temporal modeling module of TSM. However, FAN still obtains better performance compared with Separated Training (ST) at all frames and achieves the highest accuracy at 24 Frame, owing to the design of Mutual Distillation.

A.3 DIFFERENT MIDDLE SEQUENCES

Table 3: Experiments with different middle sequences on Something-Something V1.

Method	v^M	Top-1 Acc.(%)						
		4 Frame	6 Frame	8 Frame	10 Frame	12 Frame	14 Frame	16 Frame
TSM Lin et al. (2019)	-	20.60	30.23	37.36	42.72	45.97	47.49	48.55
TSM-ST	-	39.71	43.73	45.63	47.31	47.71	48.01	48.55
TSM-FAN	8F	42.85	46.57	48.20	48.81	48.90	50.47	50.79
TSM-FAN	10F	43.10	44.77	47.81	49.26	49.63	50.67	51.12
TSM-FAN	12F	42.92	43.57	46.82	48.85	49.73	50.40	50.79

Another design choice in our method is the selection of middle sequence v^M, as v^L and v^H are usually set at first based on the range of the computations. Thus, we sample 8/10/12 frames for v^M respectively and evaluate them at various frame numbers in Table 3. When we sample 8 frames for v^M, FAN obtains the best performance at 8 frames compared to the other two choices and the phenomenon is the same when sampling 10 or 12 frames for v^M. This meets our expectation as the specialized normalization for v^M learns its corresponding transformation. Overall, all the three choices leads to consistent improvement over Separated Training (ST) at all frames.

A.4 DIFFERENT DEPTHS

As we have shown in Figure 1, Temporal Deviation exists in different depths of the network which means it has no relation to the representation ability. But can FAN address this issue at other depths? As previous experiments are built on ResNet-50 He et al. (2016), we conduct experiments on ResNet-18, ResNet-101 and include their results in Table 4. The results prove that FAN can resolve Temporal Deviation regardless of the depths of the deep network.

A.5 OUT-OF-BOUNDS RESULTS

In previous sections, we only evaluate our method within the range of v^L(4 Frame) and v^H(16 Frame). Though FAN can handle most of the cases if we set the range to be as large as possible, we are still curious about the performance of FAN when it is validated at frames which are out of the

Table 4: Experiments with different depths on Something-Something V1.

Method	Top-1 Acc.(%)		
	4 Frame	8 Frame	16 Frame
TSM(R18) Lin et al. (2019)	16.82	33.12	42.95
TSM(R18)-ST	32.33	38.21	42.95
TSM(R18)-FAN	**36.83**	**41.61**	**43.57**
TSM(R101) Lin et al. (2019)	22.15	39.30	49.57
TSM(R101)-ST	40.76	46.96	49.57
TSM(R101)-FAN	**45.15**	**50.24**	**51.79**

Table 5: Evaluation at out-of-bounds frames on Something-Something V1.

Method	Top-1 Acc.(%)				
	2 Frame	4 Frame	8 Frame	16 Frame	20 Frame
TSM-ST	26.79	39.71	45.63	48.55	47.69
TSM-FAN	**26.91**	**42.85**	**48.20**	**50.79**	**50.42**

range. Therefore, we evaluate FAN at frame 2/20 and compare it with Separated Training in Table 5. FAN exhibits similar performance at 2 Frame compared to ST and show obvious advantage at 20 Frame. The reason is that TSM with 20 frames has lower performance compared with sampling 16 frames which we have explained in Section A.1. While the accuracy of FAN with 20 Frame only drops slightly compared to 16 Frame which demonstrates the robustness of our design.

A.6 Quantitative Results

In the Empirical Validation Section, we show our results in the figure and we also provide the quantitative results in the following tables for references:

Table 6: Experiments with different architectures on Something-Something V1.

Method	Top-1 Acc.(%)		
	v_L	v_M	v_H
TSM Lin et al. (2019)	20.60	37.36	48.55
TSM-ST	39.71	45.63	48.55
TSM-FAN	**42.85**(3.14↑)	**48.20**(2.57↑)	**50.79**(2.24↑)
TEA Li et al. (2020b)	21.78	41.49	51.23
TEA-ST	41.36	48.37	51.23
TEA-FAN	**44.97**(3.61↑)	**51.61**(3.24↑)	**54.04**(2.81↑)
SlowFast Feichtenhofer et al. (2019)	15.08	35.08	45.88
SlowFast-ST	39.91	44.12	45.88
SlowFast-FAN	**43.90**(3.99↑)	**47.11**(2.99↑)	**47.27**(1.39↑)
Uniformer Li et al. (2022)	19.02	41.64	51.61
Uniformer-ST	41.46	45.26	51.61
Uniformer-FAN	**44.96**(3.50↑)	**51.02**(5.76↑)	**53.99**(2.38↑)

Table 7: Experiments with different datasets on TSM.

Method	Dataset	Top-1 Acc.(%)		
		v_L	v_M	v_H
TSM Lin et al. (2019)		31.52	51.55	61.02
TSM-ST	Sth-Sth V2	53.38	59.29	61.02
TSM-FAN		**56.07**(2.69↑)	**61.86**(2.57↑)	**63.61**(2.59↑)
TSM Lin et al. (2019)		64.10	69.77	73.16
TSM-ST	Kinetics400	66.25	70.38	73.16
TSM-FAN		**68.96**(2.71↑)	**72.33**(1.95↑)	**74.35**(1.19↑)
TSM Lin et al. (2019)		42.16	46.38	48.30
TSM-ST	HMDB51	44.74	46.77	48.30
TSM-FAN		**45.67**(0.93↑)	**47.67**(0.90↑)	**48.80**(0.50↑)