This CVPR Workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Efficient VideoMAE via Temporal Progressive Training

Xianhang Li[1] Peng Wang[2] Xinyu Li[3] Heng Wang[4] Hongru Zhu[5] Cihang Xie[1]
[1]UC Santa Cruz [2]Bytedance [3]Amazon [4]Nvidia [5]JHU

Abstract

Masked autoencoders (MAE) have recently been adapted for video recognition, setting new performance benchmarks. Nonetheless, the computational overhead of training VideoMAE remains a prominent challenge, often demanding extensive GPU resources and days of training. To improve the efficiency of VideoMAE training, this paper presents Temporal Progressive Training (TPT), a simple yet effective method that strategically introduces longer video clips along the training process. Specifically, TPT decomposes the intricate task of long-clip reconstruction into a series of incremental sub-tasks, progressively transitioning from short to long video clips. Our extensive experiments demonstrate the efficacy and efficiency of TPT. For example, TPT reduces training costs by factors of $2\times$ *on Kinetics-400 and* $3\times$ *on Something-Something V2, while maintaining the performance of VideoMAE. Furtherore, when given the same training budget, TPT consistently surpass VideoMAE by 0.4-0.5% on Kinetics-400 and 0.2-0.6% on Something-Something V2.*

1. Introduction

Self-supervised pre-training has rapidly emerged as a prevailing paradigm for large-scale representation learning in both natural language processing and computer vision. Within this paradigm, Masked Autoencoder (MAE) stands as a representative approach, inspired by Masked Language Modeling (MLM), which masks random portions of input tokens and trains models to reconstruct the masked tokens. MAE offers simplicity, effectiveness, and robustness across both image and video domains [17, 21, 43]. However, training MAE models, especially in the context of video recognition, presents a substantial computational challenge, thereby limiting its accessibility for researchers with constrained resources. For instance, training a standard VideoMAE model on Kinetics-400 necessitates a massive investment of up to 5.6 days using 64 GPUs.

Efficiency in learning inherently rests on selecting informative samples and reducing redundancy, a principle well-corroborated in prior research on image-based learning [23, 31]. However, the current design of VideoMAE overlooks some of these considerations — for example, VideoMAE directly learns from joint spatiotemporal samples, such as cuboid-shaped data with inherent redundancies. While VideoMAE employs strategies like aggressive masking in the spatial dimension to expedite training, we posit that considerable opportunities remain untapped to further optimize its training speed by addressing the inefficiencies associated with the temporal dimension.

In this paper, we present *Temporal Progressive Training* (TPT), a method designed to reduce computational overheads by mitigating the temporal redundancy in VideoMAE training. The key idea of TPT is to introduce a strategic "warm-up" approach to the intricate task of long video reconstruction. As illustrated in Figure 1, TPT decomposes the video reconstruction process into a series of incremental sub-tasks, initiating with short video snippets and gradually extending to full-length clips. To balance efficiency and efficacy, we undertake an in-depth examination of various aspects of this task decomposition strategy, including 1) the design specifics of sub-tasks, (*e.g.*, sub-task objectives, sub-task counts, frame length and allocated computation in each sub-task) and 2) the the optimal training recipe (*e.g.*, batch size, learning rate schedule). A key insight from our investigation is that while identifying the optimal configurations for TPT requires additional effort, this calibration can be achieved using a relatively small-scale dataset with a limited training budget. Once identified, these configurations generalize effectively across a variety of datasets and models.

To evaluate the efficiency and effectiveness of TPT, we conduct extensive experiments on the Something-Something-V2 [19] and Kinetics-400 [6] datasets. Our empirical results demonstrate that, within a given training budget, TPT consistently outperforms VideoMAE [17, 43] by a notable margin (0.4-0.5% on K400, 0.2-0.6% on SSV2). Additionally, TPT can competitively match the performance of VideoMAE using approximately half the training expense for Kinetics-400 and only one-third for Something-Something-V2. We hope this work serves as a strong benchmark and catalyzes further research into efficient video training.

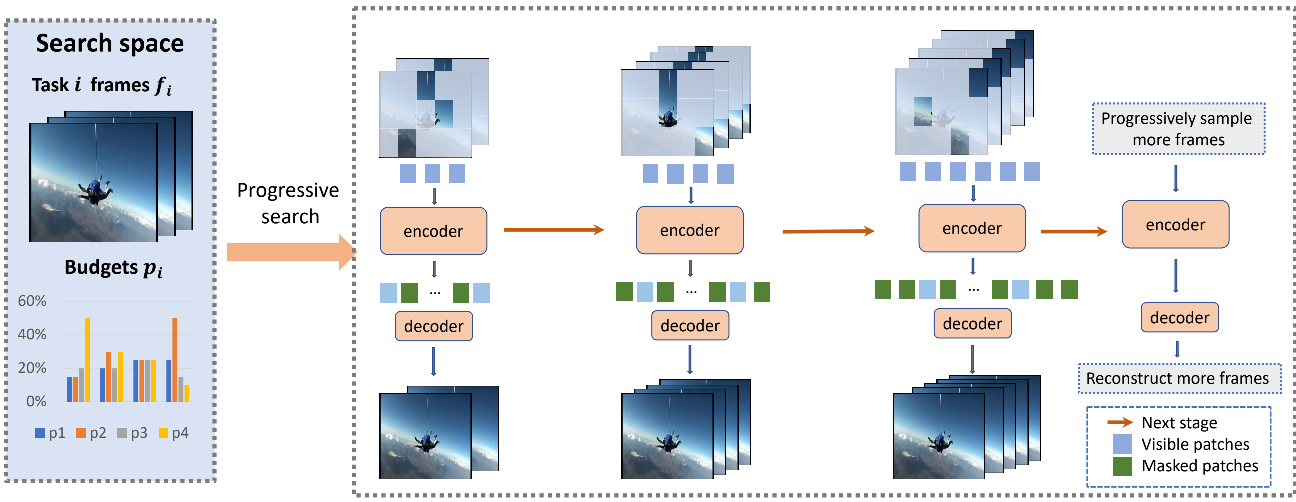


Figure 1. **Overview of our Temporal Progressive Training (TPT) pipeline.** It is a multi-stage VideoMAE training framework. We set up a search space of TPT (left), which is then applied in training (right). It starts the training with fewer frames for spatial semantics learning, and then progressively increases the temporal length to learn at each stage. Finally, the model learns the full spatiotemporal representations.

2. Related Work

Video recognition. Video recognition often adapts 2D image classification models, such as ResNet [20], VGG [37] and Inception [41], to serve as effective backbones. Enhancements to these models include video-specific improvements to the 2D backbones [22, 25, 26, 28, 30, 32, 39], attention-style architectures [5, 8, 48, 49, 53, 56], and the adoption of 3D CNN backbones [6, 15, 16, 24, 27, 44]. Recently, Vision Transformer (ViT) [12] have advanced video understanding, with large-scale pre-training significantly improving performance [1, 3, 4, 13, 14, 18, 33, 35, 36, 46, 54, 55]. However, such pre-training paradigm often comes with substantial computational overheads. In this paper, our work explores the efficiency potential of progressive spatiotemporal pre-training. As part of our future work, we plan to investigate the effectiveness of the proposed temporal progressive training on dense video prediction tasks, including novel video segmentation benchmark[10, 11], to further validate its generalizability and robustness.

Masked-autoencoder. The masked autoencoding paradigm, well-documented in natural language processing [9], has demonstrated exceptional representation learning during pre-training. This success has extended to computer vision, with iGPT pioneering image pixel prediction [7]. ViT [12] further builds on top by predicting unseen image patches in a self-supervised manner, while BEIT [2] extends the paradigm to discrete token prediction. MAE [21] innovatively optimizes the patch masking strategy to significantly reduce computations, inspiring extensions like BEVT and VIMPAC for discrete token prediction [42, 47]. OmniMAE enables label-free training across images and videos [13], and VideoMAE and ST-MAE showcase the adaptability of MAE to data-limited scenarios [17, 43]. Our research aligns with these trends, and specifically focusing on enhancing spatiotemporal pre-training efficiency.

3. Method

In this section, we detail our proposed Temporal Progressive Training. We begin by revisiting ImageMAE [21] and VideoMAE [43] as preliminaries in Section 3.1. In Section 3.2, we elaborate on the framework of TPT and its formulation, followed by the optimal training recipe and the scaling behavior of TPT.

3.1. Revisiting ImageMAE and VideoMAE

ImageMAE [21] employs an asymmetric encoder-decoder structure with a masking-then-predicting paradigm, randomly masking out 75% of grid patches for reconstruction. Later works [17, 43] such as VideoMAE generalize this approach to video recognition by masking 90% of grid cuboids in a video clip for reconstruction.

VideoMAE input. Consider raw video data $\mathbf{X} \in \mathbb{R}^{C\times T\times H\times W}$ with input image channel C and spatiotemporal resolution $T \times H \times W$. Videos are first temporally downsampled by a factor of τ, yielding low-fps video $eX' \in \mathbb{R}^{C\times T/\tau\times H\times W}$. A training batch comprises B randomly sampled video clips, each containing t frames with a spatial resolution $h \times w$. VideoMAE then takes input ex of size $B\times C\times t\times h\times w$.

VideoMAE encoder. The network first applies a cube embedding, using a non-overlapping 3D convolution to transform $\boldsymbol{ex}$ into ViT input tokens $\boldsymbol{et}$ of size $B \times C_p \times (\frac{t}{s_t} \times \frac{h}{s_h} \times \frac{w}{s_w})$, where C_p is the embedding dimension, and s_t, s_h, s_w are the temporal and spatial downsampling stride. With $h = w$ and $s_h = s_w$ in ViT, we denote $p = \frac{h}{s_h} = \frac{w}{s_w}$ and $t' = \frac{t}{s_t}$. A tube mask operation masks a large portion of tokens $\boldsymbol{et}$ within the spatial dimension across all frames t'. Thus, the encoder processes visible tokens of size $B \times C_p \times (t' \times p \times p \times \rho)$, where ρ is the sampling rate (*e.g.*, 10% in VideoMAE).

VideoMAE decoder. VideoMAE decoder duplicates a mask token to fill the unseen token in the corresponding position, outputting data of the same size as input. Finally, a mean squared error (MSE) loss is applied; after pre-training, the learned weights are used to initialize downstream fine-tuning in tasks such as video classification.

3.2. Temporal Progressive Training

While VideoMAE employs aggressive masking to reduce redundancy in the spatial dimension, the temporal dimension remains largely untapped. Temporal redundancy, inherent in video data due to the high similarity between consecutive frames, presents a substantial opportunity for further optimization. Our TPT is designed to exploit this potential with a simple yet effective “warm-up” approach. As shown in Figure 1, TPT decomposes full-length video reconstruction into a cascade of structured sub-tasks. Each sub-task focuses on reconstructing video clips of progressively increasing duration. Such progressive training ensures that early stages with shorter clips are computationally cheaper, leading to faster convergence and more efficient use of resources. This not only accelerates the training process but also allows for better resource allocation, making high-quality video recognition more accessible.

Specifically, to explore a universally applicable configuration for training VideoMAE models with TPT, we consider two key factors—sub-tasks design specifics (see Section 3.2.1) and the training recipe (see Section 3.2.2). The design of sub-tasks play a pivotal role in balancing efficiency and downstream performance. We evaluate the potential advantages of initiating with simpler tasks and sequentially advancing to more complex ones, and analyze the number of sub-tasks and the allocation of computational resources to each. The other vital factor is the training recipe for each sub-task. We primarily focus on batch size (bz_i) and learning rate (lr_i). Optimized training recipe ensures a smooth transfer of knowledge from one sub-task to the next.

3.2.1. Sub-task design specifics

Type of sub-task. We define a sub-task as a component of VideoMAE training focused on reconstructing a video at a specific length. As a special case, spatial image reconstruction is natually the smallest sub-task with 1 frame. Our video reconstruction sub-tasks can be designed based on a sampling sequence, either arbitrary or following a progressive order (*e.g.*, $2 \rightarrow 4 \rightarrow 8 \rightarrow 16$ frames).

Number of sub-tasks N. We incrementally vary the number of sub-tasks N starting from a baseline value of 1. We define our search space for N as $\mathbb{S} = \{1, 2, 3, 4, 5\}$. Concurrently, we consider the number of frames allocated for each sub-task, denoted as f_i, drawing from a predefined set $\mathbb{F} = \{2, 4, 8, 12, 16\}$. For instance, a configuration of N=3 and $f_i = \{2, 8, 16\}$ implies the application of 2 frames in the first stage, 8 frames in the second, and 16 frames in the final stage. We observe that training with increased number of sub-tasks typically outperform those with one sub-task or two sub-tasks (*e.g.*, $2 \rightarrow 16$). However, performance plateau beyond a certain threshold, typically $N \geq 4$.

Computation budget p_i of sub-task. Optimizing the allocation of computations across sub-tasks is crucial to balance efficiency and pre-training performance. For instance, while a shorter video reconstruction sub-task may enhance training efficiency, excessive computations on this sub-task falls short in boosting overall classification performance compared to longer video reconstruction.

Our strategy for computation allocation operates on the total floating-point operations (FLOPs). We first denote the total training budget as P_{total}, reflecting the computational requirements of the full video reconstruction task. In a conventional 1-stage VideoMAE training, we have

$$P_{total} = N_e \cdot \Phi \cdot f,$$

where N_e is total number of epochs, Φ is FLOPs for a single frame and f is number of frames in full-length video clips. Here for illustration, we would keep the total computation budget the same while decomposing it into sub-tasks.

We use different allocation ratios r_i to tailor allocated training budgets p_i for different sub-tasks. Budge allocation ratio r_i for each sub-task i is constrained by:

$$\sum_i (r_i \cdot P_{total}) = \sum_i p_i = P_{total}.$$

To further determine the number of training epochs N_{ei} for each sub-task i, we express N_{ei} as

$$N_{ei} = \frac{p_i}{\Phi \cdot f_i},$$

where p_i is the budget for sub-task i and f_i is the number of frames for this sub-task. We can reformulate N_{ei} into

$$N_{ei} = \frac{p_i}{\Phi \cdot f_i} = \frac{r_i \cdot P_{total}}{\Phi \cdot f_i} = r_i \cdot N_e \cdot \frac{f}{f_i}.$$

This allows us to calculate precise training epochs assigned to a sub-task by applying a proportional budget r_i. For instance, consider a case where we decompose a conventional VideoMAE training with target-frame $f = 16$ and total number of epochs $N_e = 800$. If we aim to maintain the total computation budget and allocate $r_i = \frac{1}{4}$ of the budget for a sub-task with frame $f_i = 2$, we should assign $N_{ei} = \frac{1}{4} \cdot 800 \cdot \frac{16}{2} = 1600$ epochs for this sub-task.

Adopting this strategy, we standardize total computation for each variant, allowing for direct performance comparisons driven by the allocation of resources, separate from the overall computation. In practice, we observe that allocated budgets can be substantially larger for computation-efficient sub-tasks with 2 or 4 frames compared to computation-heavy sub-tasks with 16 frames. Consequently, we can allocate more computation resources to more computation-efficient tasks without sacrificing overall performance, resulting in a more efficient training process.

3.2.2. Progressive training recipe

Ensuring the knowledge learned in one sub-task can be smoothly transferred to the next is paramount in progressive training. In search for optimal progressive training recipe, we focus on learning rates and batch sizes in sub-tasks.

The initial learning rate for transitioning between subtasks is a critical hyperparameter that directly influences model convergence and overall performance. It is particularly important in the case of multiple sub-tasks where underfitting and overfitting risks are magnified. The choice of optimal initial learning rates is compounded by the relatively limited number of frames in each sub-task.

A well-curated batch size is essential to achieve precise balance between resource efficiency and model efficacy. Increasing the batch size (bz_i) can significantly curtail the total consumption of GPU hours, thereby enhancing computational efficiency. However, adjusting the batch size has profound implications on the performance of the downstream task — larger batch sizes may affect the model's generalization capabilities and the accuracy of gradient estimates during optimization.

To address these challenges, we conduct a parameter search specifically for our training ablation experiments using VideoMAE-base-16f model on the Something-Something V2 dataset, which comprises approximately 220,000 video clips. To control our training budget, we limited the process to 400 total number of epochs, equivalent to viewing 0.14 billion video frames. This search aims to optimize learning rates and batch sizes to ensure smooth transitions between sub-tasks while maintaining high performance. When scaling up training schedule, model size, and dataset, we maintain these optimized recipe to ensure consistency and reliability in our training process.

3.2.3. Schedule, model and data scaling

After finalizing the optimal training recipe with base-size model on Something-something V2 dataset for 400 epochs as detailed above, we proceed with the recipe for further schedule, model and dataset scaling.

Schedule scaling First we explore scaling up training schedule, increasing the total number of epochs N_e from 400 to 800 and 1600. With the total number of epochs and computational budget fixed, there are numerous ways to scale up schedule across different sub-tasks. For example, training for 400 epochs on a 2-frame sub-task consumes the same amount of computational budget compared to training for 50 epochs on a 16-frame sub-task. Thus, we resort to a simple strategy that has been proven effective—multiplying each sub-task epoch N_{ei} by a fixed factor of 2 or 4 when scaling from $N_e = 400$ to $N_e = 800$ or $N_e = 1600$ respectively.

Model scaling and data scaling Subsequently, for model scaling, we further expand the base model to large and huge configurations. For dataset scaling, we utilize the Kinetics-400 dataset. Note that we do not conduct further search experiments across different model and dataset configurations, given that our initial training recipe remains effective after scaling. Remarkably, this underscores the adaptability of our TPT strategy, which is effortlessly applicable across various ViT variants and sizes, and different datasets.

The scalability of our TPT recipe avoids the excessive computational cost typically associated with the search processes, advocating for its adoption as a cost-efficient and highly effective training strategy, which we will further demonstrate in subsequent sections.

4. Implementation

Datasets. To validate the effectiveness of our method, we conduct experiments on two large-scale video recognition datasets: Kinetics-400 (K400) [6] and Something-Something V2 (SS-V2) [19].

K400 comprises approximately 260K raw videos, each categorized into one of 400 action categories. For K400 pre-training and fine-tuning, we adopt a fixed-stride sampling strategy widely used in the literature [15, 16, 49]. SS-V2 consists of about 220K videos, each showcasing one of 174 predefined human-object interactions with everyday objects. To handle the short videos in SS-V2 (*i.e.*, 3 seconds per video), we employ a segments-based sampling method [45]. To ensure fair comparisons between different experiments, we set the spatial resolution of all input frames to 224 $\times$ 224 by default. During pre-training, we vary only the temporal dimension t.

Training and evaluation. Unless otherwise noted, we use the following settings as our default for pre-training. We utilize the AdamW optimizer [34] with a weight decay of

type of sub-task	sampling order	Top-1
Fixed Frames Within Sub-tasks	(2 → 4 → 8 → 16)	**69.4**
Variable Frames Within Sub-tasks		68.8
Fixed Frames Within Sub-tasks	randomly sampling	68.9
Variable Frames Within Sub-tasks		68.4

Table 1. Type of sub-task. Upper, for each sub-task, we can choose to use long-cycle or short-cycle style. Bottom, for each sampling, we can fix the order progressively or randomly. Progressive sampling and a single resolution for each stage work the best.

# of sub-tasks	# of frames per task	Top-1
1	$16 \times$ s	68.4
2	$(2, 16) \times$ s	69.0
3	$(2, 4, 16) \times$ s	69.1
3	$(2, 8, 16) \times$ s	69.2
3	$(4, 8, 16) \times$ s	69.0
4	$(2, 4, 8, 16) \times$ s	**69.4**
5	$(2, 4, 8, 12, 16) \times$ s	69.1

Table 2. Ablation on the number of tasks. 's' denotes our temporal sampling strategy is segments-based [45].

2-f	4-f	8-f	16-f	Training FLOPs	Top-1
0	0	0	100%	35.5G	68.4
12.5%	6.25%	6.25%	75%	34.0G	69.2
12.5%	6.25%	31.25%	50%	33.0G	69.0
12.5%	12.5%	50%	25%	31.9G	69.4
12.5%	25%	50%	12.5%	**31.1G**	**69.4**
25%	31.25%	31.25%	12.5%	30.7G	69.3
50%	25%	12.5%	12.5%	30.3G	68.7
75%	6.25%	6.25%	12.5%	29.4G	68.2
100%	0%	0%	0%	28.8G	66.7

Table 3. **Ablation on allocating computation.** Training FLOPs are measured by averaging model GFLOPs (floating-point operations, in # of multiply-adds $\times 10^9$) via allocated training budgets.

0.05. We adopt a cosine decay learning rate scheduler with an initial learning rate of $1.5e^{-4}$ and warmup epochs of 40, as suggested in [43]. We use a batch size of 1024 on 64 Tesla-V100 GPUs for all pre-training tasks, namely a consistent batch size of 16 per GPU.

During fine-tuning, we adopt a full-model fine-tune scheme to establish a fair comparison following previous work [17, 21, 43, 50].

During evaluation, we use multi-crop multi-view protocol as suggested in [17, 43, 50]. We use 3 uniformly sampled spatial views for the SS-V2 dataset and 5 views for K400 dataset for a fair comparison.

5. Experiment Results

In Section 5.1, we begin by ablating the sub-task design specifics and the training recipe in our TPT, In Section 5.2, we present our main results after scaling up the model size and the training schedule on K400 and SS-V2 dataset. We compare our TPT framework with state-of-the-art methods in Section 5.3.

5.1. Ablation study

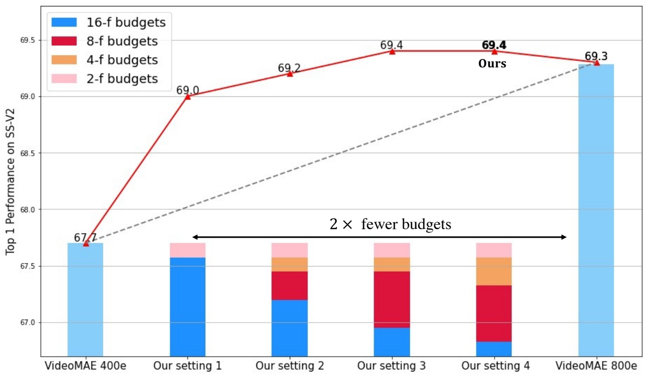
Our ablation studies are conducted on SS-V2 wth the ViT-Base [43] model as the backbone. We employ total training budgets of 400 epochs with target frames of 16. Unless otherwise specified, we report the Top-1 accuracy after fine-tuning for 30 epochs on the same dataset. We follow the evaluation setup outlined in Section 4.

Type of sub-task. We explore different types of sub-tasks by comparing two styles: (1) the number of frames to reconstruct is fixed within each sub-task but varies across sub-tasks, vs (2) the number of frames to reconstruct can change within each sub-task but all sub-tasks are the same.

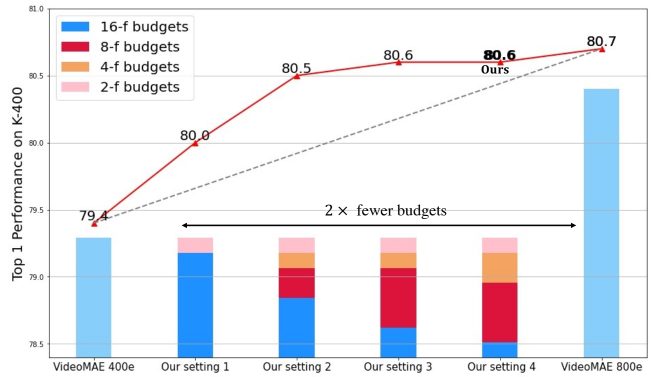
(1) *Fixed Frames Within Sub-tasks*: Each sub-task uses a fixed number of frames. For example, we can start with a sub-task of 2-frame reconstruction, then move to a sub-task of 4-frame reconstruction, followed by another sub-task of 8-frame reconstruction, and finally a sub-task of 16-frame reconstruction. Each sub-task exhausts its allocated computation budget before changing the number of frames.

(2) *Variable Frames Within Sub-tasks*: In contrast, this approach allows the number of frames to change within each sub-task, but all sub-tasks are the same. For example, within the first sub-task, we perform a cycle of frame reconstructions: 2 frames, 4 frames, 8 frames, and finally 16 frames, exhausting computation budget assigned to the first sub-task. Then in the next and all subsequent sub-tasks we repeat the change of frame numbers in the exact order.

Additionally, we further compare the progressive sampling order for the number of frames to reconstruct, e.g., $(2 \rightarrow 4 \rightarrow 8 \rightarrow 16)$, with the random temporal order, e.g., $(4 \rightarrow 2 \rightarrow 16 \rightarrow 8)$. As shown in Table 1, *Fixed Frames Within Sub-tasks* yields better performance. Furthermore, the performance using the progressive sampling order surpasses that using the random temporal order. Consequently, we adopt fixed frames within sub-tasks with a progressive sampling order for the number of frames to reconstruct across sub-tasks.



(a) **SS-V2**



(b) **K-400**

Figure 2. Allocate budgets. We use 400-epoch total FLOPs as training budgets for both datasets. Each bar refers to one allocation setting. Each point in the red line represents the Top-1 performance under given training budgets. The first and last bars are our baseline methods with VideoMAE.

# of learning rate cycles	decay type	Top-1
1	cosine	68.7
2	cosine	69.1
4	cosine	**69.4**
4	step	68.3

Table 4. Learning rate scheduler for each sub-task. We ablate the choice of decay strategy and a number of decay cycles. For each stage, a full decay cycle works best, and we choose cosine decay by default.

2-*f*	4-*f*	8-*f*	16-*f*	speedup	Top-1
8	8	8	16	× 1.0	69.4
16	16	16	16	× 2.0	**69.4**
32	32	32	16	× 2.3	69.1
64	64	64	16	× 2.4	68.8

Table 5. Batch size for each stage. Different from supervised training [51], a larger batch size can degrade the performance in our setting. Hence we choose to keep batch size the same for every stage. Here, we use per-GPU batch size for ablation.

Number of sub-tasks. We next investigate the influence of the number of sub-tasks. We allocate training resources evenly to each sub-task, with an initial task dedicated to 2-frame reconstruction. As shown in Table 2, increasing the number of sub-tasks enhances performance. Notably, performance gains reach a saturation point when utilizing the training scheduler for 4 sub-tasks in the progressive order ($2 \rightarrow 4 \rightarrow 8 \rightarrow 16$).

Allocating computation. We further vary the training computation allocations for each sub-task to examine the trade-off between efficiency and effectiveness. As presented in Figure 2, we observed that despite reduction in training FLOPs and training acceleration, allocating more budget to the initial training sub-task, which primarily focused on intensive spatial reconstruction, hinder the performance. Conversely, allocating a majority of the training budget to later sub-tasks improved performance with a notable increase in training FLOPs. Thus, our practical strategy involves directing most computational resources towards efficient sub-tasks like 2-frame or 4-frame reconstruction, with a smaller portion allocated for computation-heavy sub-tasks like the 16 frames. This strategy optimizes training efficiency without sacrificing overall performance. Therefore, we allocate computation resources across each stage as follows: 12.5%, 25%, 50%, and 12.5%, respectively.

Learning rate scheduler and batch size. In Table 4 and Table 5, we present results from the search experiment for the optimal learning rate and batch size in each sub-task. We attain the best performance with a separate cosine decay scheduler in each sub-task instead of utilizing a single schedule across all sub-tasks. Additionally, we found that scaling the batch size based on the number of frames used for reconstruction shows a detrimental effect on performance. This is because that the model is more likely to converge to a local minimum with a larger batch size. Based on our empirical results, we selected a consistent batch size of 16 per gpu for all sub-tasks, equivalent to a batch size of 1024 on 64 GPUs.

Backbone Finally, we ablate the backbone for our Temporal Progressive Training (TPT-MAE). First, as shown in Table 6, our TPT-MAE surpasses other methods with the

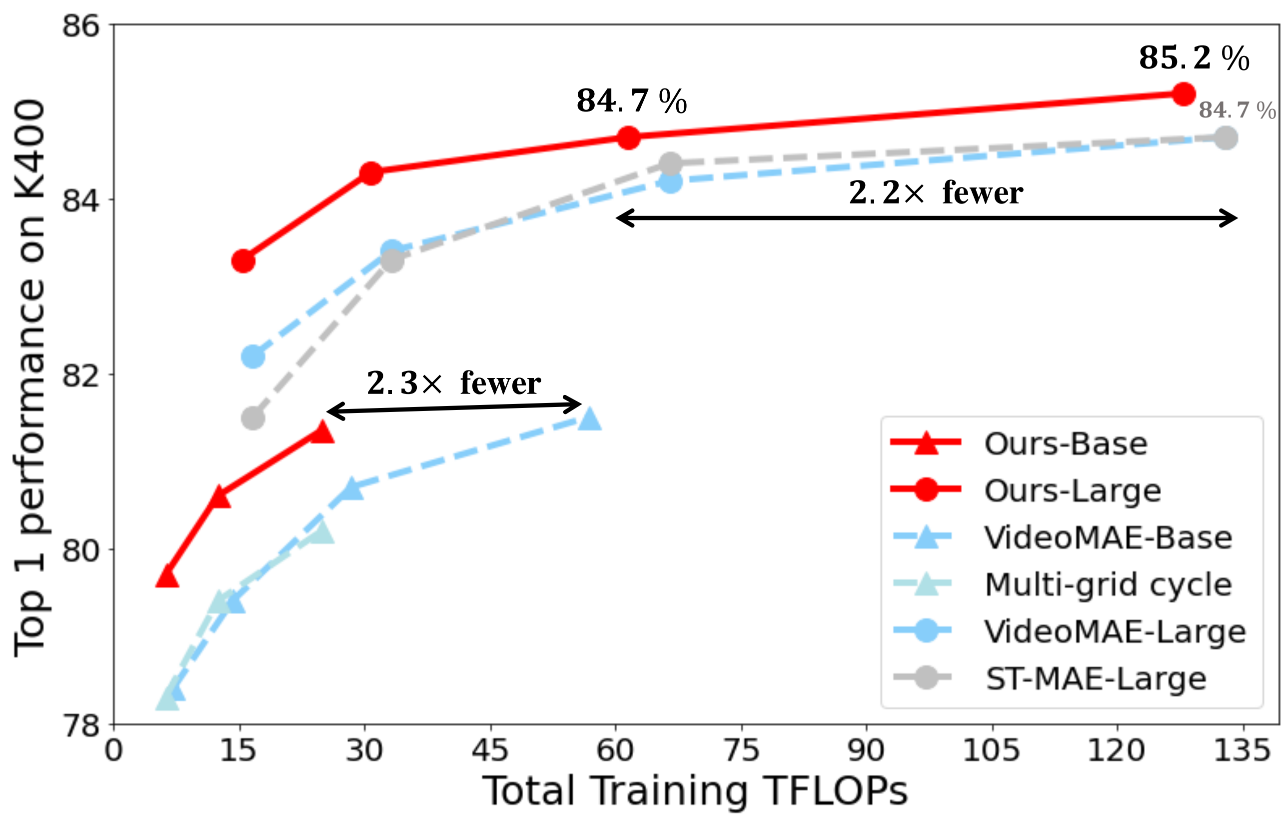


Figure 3. **Accuracy/Training cost trade-off curve on Kinetics-400 dataset [6] with ViT [43] model series.** Each point corresponds to a model with a specific pre-training budget, and we report fully fine-tuning results. Our proposed Temporal Progressive Training framework achieves a preferable trade-off between efficiency and performance.

Method	Source data	Target	Top-1
ImageMAE [21]	ImageNet-1k	pixel	64.5
VideoMAE [43]	SS-V2	pixel	67.9
MotionMAE [52]	SS-V2	RGB-diff.+pixel	68.4
MAM^2 [38]	SS-V2	RGB-diff.+token	69.0
M^{3}Video [40]	SS-V2	trajectory	69.2
TPT-MAE	SS-V2	pixel	**69.4**

Table 6. **Comparison between recent VideoMAE-based methods on SS-V2**. Comparison of TPT-MAE with other SoTA MAE-based methods with a 400-epoch budget and ViT-B model. RGB-diff refers to temporal differences between frames.

Model	Method	Top-1
VideoMAE-16f [43]	ImageNet-21k sup.	61.8
	VideoMAE	67.6
	TPT-MAE	**69.3**
Timesformer-8f [3]	ImageNet-21k sup.	59.5
	VideoMAE	66.3
	TPT-MAE	**67.7**
MotionFormer-8f [36]	ImageNet-21k sup.	66.5
	VideoMAE	66.2
	TPT-MAE	**67.7**

Table 7. **Comparison of different backbones on SS-V2.** We compare with VideoMAE under a 400-epoch pre-training computation budget. 8/16f refers to the number of frames.

same MAE backbone, demostrating the effectivenss of TPT for spatiotemporal learning. Furthermore, we examine the use of different backbone architectures with our TPT framework, and find that the VideoMAE-16f backbone performs best among all alternative backbones (see Table 7).

Next, we scale up our model and the training schedule on both SS-V2 and K400 dataset. This extension enables us to compare with state-of-the-art techniques, focusing on both efficiency and performance.

5.2. Main Results

Following the scaling strategy outlined in Section 3.2.3, we conduct evaluation on larger datasets with larger models. On data and schedule scaling, we validate the effectiveness of our sub-task design and training recipe on the larger-scale

Training Method	Backbone	Pre-train on	param.	GFLOPs	input size	SSV2	K400
supervised	Slowfast [16]	K400	53M	106	32×224^2	63.1	79.8
supervised	MViT-B [14]	K400	37M	455	64×224^2	67.7	81.2
supervised	MViTv2-B [29]	K400	51M	225	32×224^2	70.5	82.9
supervised	MotionFormer [36]	IN-21K+K400	109M	370	32×224^2	66.5	80.2
supervised	Swin-B [33]	IN-21K+K400	88M	321	32×224^2	69.6	81.1
supervised	TimeSformer [3]	IN21K	430M	5549	64×224^2	62.4	80.7
MaskFeat [50]	MViTv2-L [29]	K400	218M	2828	40×224^2	74.4	84.3
VideoMAE [43]	ViT-L	K400	304M	598	16×224^2	74.0	84.7
ST-MAE [17]	ViT-L	K400	304M	598	16×224^2	72.1	84.8
TPT-MAE	ViT-L	K400	304M	598	16×224^2	**74.2**	**85.2**
ST-MAE [17]	ViT-H	K400	632M	1193	16×224^2	74.1	85.1
TPT-MAE	ViT-H	K400	632M	1193	16×224^2	**74.7**	**85.5**

Table 8. **Comparisons to previous methods on K400 and SS-V2**. We pre-train all the models for 1600 epochs on K400 and conduct full fine-tuning. On SS-V2, following the ST-MAE [17], we initialize our model with K400 pre-trained weights without intermediate fine-tuning. We report the Top-1 accuracy on the validation set. Parameters are measured in millions and the input size is frame$\times$spatial resolution$\times$spatial resolution.

K400 dataset with varying schedules (400, 800, and 1600 epochs). On model scaling, we establish that our design and training recipe can be seamlessly applied to larger model sizes, such as ViT-L and ViT-H, without requiring additional adjustments.

TPT-MAE uses less compute to achieve the same level of performance. We evaluate the efficiency of our TPT framework with different training budgets and models of different sizes. We compare the training FLOPs and evaluation top1 accuracy in Figure 3. Our proposed TPT approach consistently reduces model training computation while maintaining uncompromised performance. For instance, TPT-MAE consistently reduces training epochs by at least 2$\times$, maintaining up to 7.5% reduction in training FLOPs compared to VideoMAE, while achieving similar performance levels. Notable, the results suggest a more profound positive impact from our TPT framework on larger model size with longer training schedules.

TPT-MAE is more effective given the same computational budget. We further compare TPT with other MAE models at comparable FLOPs. Our experimental results demonstrate that TPT outperforms other MAE models at different training budgets, providing a 0.5% improvement in Top-1 accuracy over the ViT-L model using the 1600-epoch setting (Table 8. These findings showcase the effectiveness of the proposed TPT, which generalizes well to various computational budgets and application scenarios.

5.3. SoTA Comparison

To further assess the effectiveness of our proposed TPT approach, we compare the performance and efficiency of our model with the state-of-the-art (SoTA) results reported in the literature in Table 8.

Kinetics-400. We pre-train our TPT-MAE on K-400 dataset for 1600 epochs for a fair comparison. Our TPT-MAE consistently improves performance of VideoMAE: +0.5% up on ViT-L, and +0.4% on ViT-H. This shows that the proposed TPT learns better spatiotemporal semantics.

Something-something V2. Our experiments on SS-V2 dataset reveal similar patterns, as TPT-MAE consistently enhances the performance of VideoMAE. Specifically, we observe performance improvements of +0.2% on ViT-L and +0.6% on ViT-H. These consistent results on both large-scale datasets indicate that the proposed TPT approach generalizes to datasets of varying sizes and characteristics.

6. Conclusion

We propose a Temporal Progressive Training framework to speed up and enhance VideoMAE. Our framework separates the learning process into low-cost spatial training and more informative-dense but costly temporal training. We begin training with fewer frame for learning spatial semantics and progressively increase the temporal clip length at each stage. More importantly, we propose efficiently searching for an optimal temporal progressive training recipe under a small training budget. As a result, the proposed TPT outperform VideoMAE [17, 43] by a notable margin under any training budget constraints. In addition, TPT requires ~2-3$\times$ less training cost to reach the SoTA performance.

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *ICCV*, 2021. 2

[2] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. In *ICLR*, 2021. 2

[3] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding. In *ICML*, 2021. 2, 7, 8

[4] Adrian Bulat, Juan-Manuel Perez-Rua, Swathikiran Sudhakaran, Brais Martinez, and Georgios Tzimiropoulos. Space-time mixing attention for video transformer. In *NeurIPS*, 2021. 2

[5] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *ICCVW*, 2019. 2

[6] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1, 2, 4, 7

[7] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020. 2

[8] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *CVPR*, 2019. 2

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2018. 2

[10] Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, and Chen Change Loy. Mevis: A large-scale benchmark for video segmentation with motion expressions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2694–2703, 2023. 2

[11] Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, Philip HS Torr, and Song Bai. Mose: A new dataset for video object segmentation in complex scenes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 20224–20234, 2023. 2

[12] 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. In *ICLR*, 2021. 2

[13] Haodong Duan, Yue Zhao, Yuanjun Xiong, Wentao Liu, and Dahua Lin. Omni-sourced webly-supervised learning for video recognition. In *ECCV*, 2020. 2

[14] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, 2021. 2, 8

[15] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *CVPR*, 2020. 2, 4

[16] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019. 2, 4, 8

[17] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. In *NeurIPS*, 2022. 1, 2, 5, 8

[18] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *CVPR*, 2019. 2

[19] 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. 1, 4

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[21] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 1, 2, 5, 7

[22] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *ICCV*, 2019. 2

[23] Tyler B Johnson and Carlos Guestrin. Training deep models faster with robust, approximate importance sampling. In *NeurIPS*, 2018. 1

[24] Dan Kondratyuk, Liangzhe Yuan, Yandong Li, Li Zhang, Mingxing Tan, Matthew Brown, and Boqing Gong. Movinets: Mobile video networks for efficient video recognition. In *CVPR*, 2021. 2

[25] Kunchang Li, Xianhang Li, Yali Wang, Jun Wang, and Yu Qiao. Ct-net: Channel tensorization network for video classification. In *ICLR*, 2021. 2

[26] Xianhang Li, Yali Wang, Zhipeng Zhou, and Yu Qiao. Smallbignet: Integrating core and contextual views for video classification. In *CVPR*, 2020. 2

[27] Xianhang Li, Huiyu Wang, Chen Wei, Jieru Mei, Alan Yuille, Yuyin Zhou, and Cihang Xie. In defense of image pre-training for spatiotemporal recognition. In *ECCV*, 2022. 2

[28] Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *CVPR*, 2020. 2

[29] 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 *CVPR*, 2022. 8

[30] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019. 2

[31] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *CVPR*, 2017. 1

[32] Zhaoyang Liu, Donghao Luo, Yabiao Wang, Limin Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Tong Lu. Teinet: Towards an efficient architecture for video recognition. In *AAAI*, 2020. 2

[33] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *CVPR*, 2022. 2, 8

[34] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2017. 4

[35] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m:

Learning a text-video embedding by watching hundred million narrated video clips. In *ICCV*, 2019. 2

[36] Mandela Patrick, Dylan Campbell, Yuki M. Asano, Ishan Misra Florian Metze, Christoph Feichtenhofer, Andrea Vedaldi, and Joáo F. Henriques. Keeping your eye on the ball: Trajectory attention in video transformers. In *NeurIPS*, 2021. 2, 7, 8

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2

[38] Yuxin Song, Min Yang, Wenhao Wu, Dongliang He, Fu Li, and Jingdong Wang. It takes two: Masked appearance-motion modeling for self-supervised video transformer pre-training. *arXiv preprint arXiv:2210.05234*, 2022. 7

[39] Swathikiran Sudhakaran, Sergio Escalera, and Oswald Lanz. Gate-shift networks for video action recognition. In *CVPR*, 2020. 2

[40] Xinyu Sun, Peihao Chen, Liangwei Chen, Thomas H. Li, Mingkui Tan, and Chuang Gan. M^3video: Masked motion modeling for self-supervised video representation learning. *arXiv preprint arXiv:2210.06096*, 2022. 7

[41] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2

[42] Hao Tan, Jie Lei, Thomas Wolf, and Mohit Bansal. Vimpac: Video pre-training via masked token prediction and contrastive learning. *arXiv preprint arXiv:2106.11250*, 2021. 2

[43] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *NeurIPS*, 2022. 1, 2, 5, 7, 8

[44] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018. 2

[45] 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. 4, 5

[46] Mengmeng Wang, Jiazheng Xing, and Yong Liu. Actionclip: A new paradigm for video action recognition. *arXiv preprint arXiv:2109.08472*, 2021. 2

[47] Rui Wang, Dongdong Chen, Zuxuan Wu, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Yu-Gang Jiang, Luowei Zhou, and Lu Yuan. Bevt: Bert pretraining of video transformers. In *CVPR*, 2022. 2

[48] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *ECCV*, 2018. 2

[49] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 2, 4

[50] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In *CVPR*, 2022. 5, 8

[51] Chao-Yuan Wu, Ross Girshick, Kaiming He, Christoph Feichtenhofer, and Philipp Krahenbuhl. A multigrid method for efficiently training video models. In *CVPR*, 2020. 6

[52] Haosen Yang, Deng Huang, Bin Wen, Jiannan Wu, Hongxun Yao, Yi Jiang, Xiatian Zhu, and Zehuan Yuan. Self-supervised video representation learning with motion-aware masked autoencoders. *arXiv preprint arXiv:2210.04154*, 2022. 7

[53] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks. In *ECCV*, 2020. 2

[54] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021. 2

[55] Bowen Zhang, Jiahui Yu, Christopher Fifty, Wei Han, Andrew M Dai, Ruoming Pang, and Fei Sha. Co-training transformer with videos and images improves action recognition. *arXiv preprint arXiv:2112.07175*, 2021. 2

[56] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *ECCV*, 2018. 2