# AIM: ADAPTING IMAGE MODELS FOR EFFICIENT VIDEO ACTION RECOGNITION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Recent vision transformer based video models mostly follow the "*image pre-training then finetuning*" paradigm and have achieved great success on multiple video benchmarks. However, full finetuning such a video model could be computationally expensive and unnecessary, given the pre-trained image transformer models have demonstrated exceptional transferability. In this work, we propose a novel method to Adapt pre-trained Image Models (AIM) for efficient video understanding. By freezing the pre-trained image model and adding a few lightweight Adapters, we introduce spatial adaptation, temporal adaptation and joint adaptation to gradually equip an image model with spatiotemporal reasoning capability. We show that our proposed AIM can achieve competitive or even better performance than prior arts with substantially fewer tunable parameters on four video action recognition benchmarks. Thanks to its simplicity, our method is also generally applicable to different image pre-trained models, which has the potential to leverage more powerful image foundation models in the future.

## 1 INTRODUCTION

The "pre-training then finetuning" paradigm has played an important role in computer vision. The key to this paradigm is a well pre-trained image model, which can provide strong transferability to downstream tasks through finetuning. Recently, large foundation models (Radford et al., 2021; Yuan et al., 2021b; Jia et al., 2021; Wang et al., 2022) can even demonstrate remarkable few-/zero-shot performance given their learned superior visual representations.

In video understanding, a common practice is also bootstrapping from an image pre-trained model and then finetuning on the video data. There are two dominating directions as shown in Fig. 1, one is to extend an image model with additional temporal module (Lin et al., 2019; Bertasius et al., 2021; Arnab et al., 2021), the other is to inflate an image model to a video model (Carreira & Zisserman, 2017; Liu et al., 2022). However, there exists at least two drawbacks for the aforementioned methods. First, most approaches require full finetuning (*i.e.*, updating all the model parameters during training) to achieve promising results on common video benchmarks. This is quite costly in terms of both computation and memory footprint, *e.g.*, 1200 Tesla V100 GPU hours to train Liu et al. (2022). Second, it also remains questionable that whether it is necessary to full finetune pre-trained image models given that they have demonstrated excellent transferability. An inadequate finetuning on downstream data might destroy the well generalized representations from such foundation models.

To overcome the drawbacks, a research direction termed parameter-efficient transfer learning has been trending in natural language processing (NLP) (Houlsby et al., 2019; Lester et al., 2021; Ben Zaken et al., 2022; Hu et al., 2022). The goal is to only finetune a small number of (extra) parameters while keeping large pre-trained language models (Devlin et al., 2018; Brown et al., 2020) frozen to attain strong performance. With the rise of large vision transformer (ViT) models, such techniques have been recently introduced to computer vision for efficient transfer learning. However, existing works either focus on tuning a pre-trained image model for image tasks (image-to-image) (Bahng et al., 2022; Jie & Deng, 2022; Jia et al., 2022), or tuning a pre-trained video model for video tasks (video-to-video) Chen et al. (2022). Directly leveraging pre-trained image models for efficient transfer learning to video tasks (image-to-video) is less explored, because image models lack the capability of temporal reasoning.

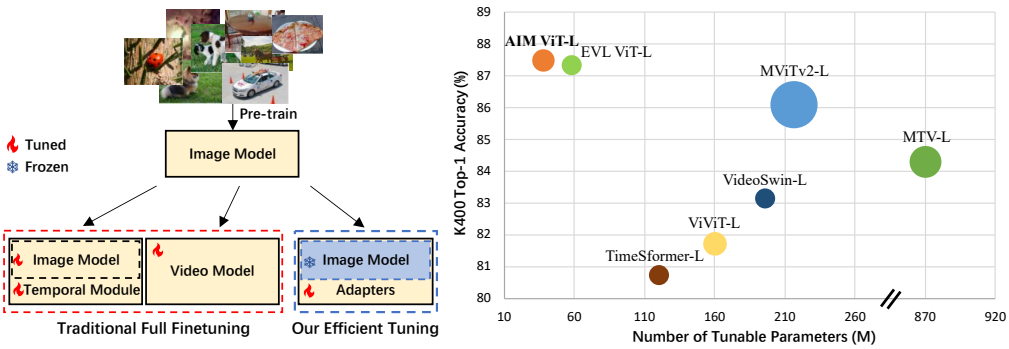


Figure 1: **Left**: Pipeline comparison between traditional full finetuning and our efficient finetuning. **Right**: Performance comparison on K400 dataset (Kay et al., 2017). Bubble size indicates GFLOPS at inference time. Our proposed AIM achieves the highest accuracy while enjoying significantly less number of tunable parameters and GFLOPS.

In this work, we introduce a new way to **A**dapt pre-trained **I**mage transformer **M**odels (AIM) for efficient video action recognition. By freezing the pre-trained image model and adding a few lightweight adapters (Houlsby et al., 2019) during finetuning, we show that our proposed AIM can achieve competitive or even better results than previous state-of-the-art methods with substantially fewer tunable parameters (Fig. 1 right). To be specific, we first introduce adapter after self-attention layer in a transformer block to perform *spatial adaptation*. We show that a well pre-trained image model is sufficiently good for spatial modeling in video understanding. Then for temporal modeling, we simply reuse the image pre-trained self-attention layer but apply it to the temporal dimension of video input, forcing it to model the relationship across different frames. An adapter is also appended for *temporal adaptation*. Finally, we perform *joint adaptation* by adding another adapter in parallel to the MLP layer in a transformer block. To summarize, we make the following contributions:

1. We propose a new way to adapt pre-trained image transformer models for efficient video understanding. Our method is generally applicable to different image pre-trained models, simple to implement, and cost-effective to train.
2. Our method is significantly more efficient than full finetuning a video model, *e.g.*, on Swin-B backbone, we can reduce the memory footprint by $50\%$ and training time by $42\%$ compared to VideoSwin (Liu et al., 2022).
3. AIM achieves comparable or higher performance than previous full finetuned state-of-the-arts on 4 video action recognition benchmarks, *e.g.*, $87.5\%$ on K400 with 38M tunable parameters.
4. Our method also brings data efficiency, *e.g.*, AIM outperforms counterpart TimeSformer (Bertasius et al., 2021) by $9\%$ absolute accuracy improvement when using $1\%$ of the training data.

## 2 Related Work

**Image pre-trained models.** ViT (Dosovitskiy et al., 2020) and its variants (Liu et al., 2021; Wang et al., 2021b; Yuan et al., 2021a; Dong et al., 2022) have been proposed to achieve state-of-the-art performance on image recognition. Once trained, these models could also serve as good initialization for transfer learning to downstream tasks. In terms of training techniques, they are commonly trained on large-scale labeled datasets (Deng et al., 2009; Sun et al., 2017; Zhai et al., 2022) in a supervised manner. To alleviate the labeling cost, self-supervised learning methods (Chen et al., 2021; Bao et al., 2021; Zhou et al., 2021; He et al., 2022b; Xie et al., 2022) are introduced to learn effective representations from unlabeled data. Recent works (Radford et al., 2021; Jia et al., 2021; Yuan et al., 2021b; Wang et al., 2022) adopt large-scale multimodal data (*e.g.*, image-text pairs) for model training, which leads to even more powerful visual representations. In this work, thanks to the simplicity of our proposed method, we could take advantage of these well pre-trained image models and adapt them efficiently to solve video tasks.

**Video action recognition.** A paradigm shift from using convolutional networks (Lin et al., 2019; Carreira & Zisserman, 2017; Tran et al., 2018; Feichtenhofer et al., 2019) to transformers has been

observed for video action recognition. Most works use image pre-trained models as initialization and extend them to video models by introducing new temporal modules (Bertasius et al., 2021; Arnab et al., 2021; Zhang et al., 2021b; Yan et al., 2022) or inflate them to video models (Liu et al., 2022). Another direction is to directly pre-train a video model in a self-supervised manner (Tong et al., 2022; Feichtenhofer et al., 2022; Wei et al., 2022; Tan et al., 2021). However, all these models are full finetuned on video data, which makes the training cost unaffordable to most researchers and practitioners. There are some recent works Ni et al. (2022); Ju et al. (2021) extending CLIP to perform action recognition, but they are multimodal methods which requires additional text branch. Our proposed AIM leverages existing pre-trained image models (no need for video model pre-training), only tunes a small number of model parameters (much more efficient than full finetuning), and achieves comparable or even better performance than previous state-of-the-arts.

**Parameter-efficient finetuning** techniques (Houlsby et al., 2019; Hu et al., 2022; Lester et al., 2021; Li & Liang, 2021; He et al., 2022a; Ben Zaken et al., 2022; Sung et al., 2021; Qing et al., 2022) are first proposed in NLP since full finetuning the increasingly larger language models for various downstream tasks becomes less feasible. Their goal is to reduce the number of trainable parameters thus lowering the computation cost, while reaching or surpassing the performance of full finetuning. Recently, parameter-efficient transfer learning is also studied in computer vision (Jia et al., 2022; Bahng et al., 2022; Chen et al., 2022; Jie & Deng, 2022). All these methods focus on adapting models in the same domain (*e.g.*, image-to-image or video-to-video), while our method adapts an image model for video tasks. One concurrent work (Lin et al., 2022) also studies how to adapt image pre-trained models for video action recognition. However, there are several major differences. First, they add new trainable decoder branches, which consist of 3D convolutions and cross-frame attention, to the frozen image encoder. We simply reuse image pre-trained self-attention to perform temporal modeling, while enjoying better performance and less tunable parameters. Second, our method is shown to be compatible with different image models, while Lin et al. (2022) only shows its effectiveness on CLIP image encoder.

## 3 METHODOLOGY

In this section, we first briefly describe ViT and video baselines (Sec. 3.1). Then we introduce spatial adaptation (Sec. 3.2), temporal adaptation (Sec. 3.3) and joint adaptation (Sec. 3.4), to show how we adapt a pre-trained image model for effective video modeling step-by-step.

### 3.1 PRELIMINARY

Since Vision Transformer (ViT) (Dosovitskiy et al., 2020) is proposed, transformer-based models have been widely adopted in various computer vision tasks, including video action recognition. In this work, we focus on adapting pre-trained image transformer models and compare to full finetuned video transformer models, unless otherwise stated.

More specifically, ViT handles an image as a sequence of small patches. Given input image $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$, ViT first splits the image to $N$ non-overlapping patches and maps each patch to a $D$-dim patch embedding via a trainable linear projection. Here, $(H, W)$ is the image resolution and $C$ is the number of channels. Patch embeddings $\boldsymbol{x}_p \in \mathbb{R}^{N \times D}$, where $N = HW/P^2$ and $P$ denotes the patch size. Then a learnable [class] token is prepended to $\boldsymbol{x}_p$ as $\boldsymbol{x}_0 = [\boldsymbol{x}_{class};\ \boldsymbol{x}_p] \in \mathbb{R}^{(N+1) \times D}$. To encode positional information, positional embeddings $\boldsymbol{E}_{pos} \in \mathbb{R}^{(N+1) \times D}$ are added to $\boldsymbol{x}_0$ as $\boldsymbol{z}_0 = \boldsymbol{x}_0 + \boldsymbol{E}_{pos}$, where $\boldsymbol{z}_0$ is the final input being fed to a sequence of transformer blocks. Each transformer block is composed of a multiheaded self-attention (MSA) and a MLP layer, together with Layernorm (LN) and skip connections, see Fig. 2(b). The computation of a standard transformer block can be written as

$$\boldsymbol{z}'_l = \boldsymbol{z}_{l-1} + \mathrm{MSA}(\mathrm{LN}(\boldsymbol{z}_{l-1})) \tag{1}$$

$$\boldsymbol{z}_l = \boldsymbol{z}'_l + \mathrm{MLP}(\mathrm{LN}(\boldsymbol{z}'_l)) \tag{2}$$

where $\boldsymbol{z}_{l-1}$ and $\boldsymbol{z}_l$ denotes the input and output of the $l$-th transformer block. Finally, the learned [class] token $\boldsymbol{x}_{class}$ from the last transformer block is used as global visual representation and fed into a classification head to make the prediction.

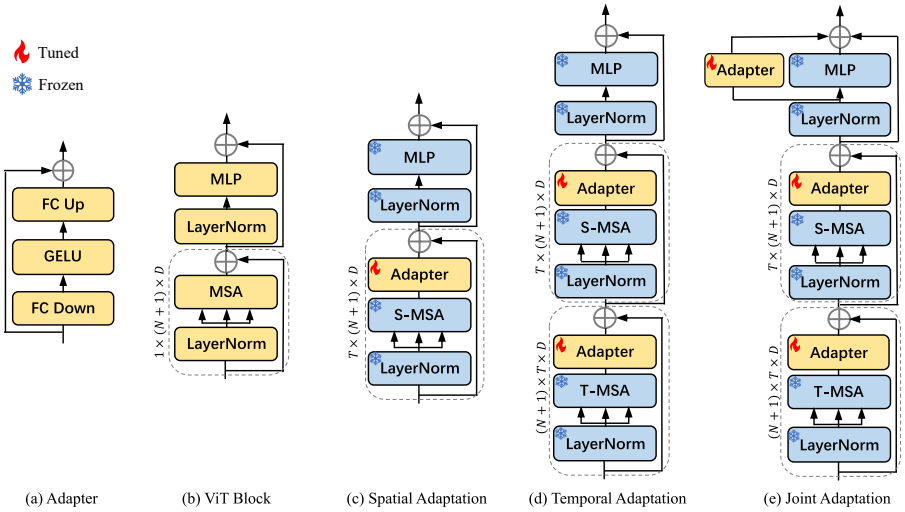


Figure 2: We show how we adapt a standard ViT block (b) for video action recognition, by gradually adding spatial adaptation (c), temporal adaptation (d) and joint adaptation (e). Note that S-MSA and T-MSA share weights but are applied to different input dimensions. During training, only newly added Adapters are updated while all the other layers are frozen.

**Space-only and space-time models for video**. A video is a stack of frames with temporal structure. Hence, video understanding requires the model to learn both good appearance representations in each frame (spatial modeling) and also infer the temporal structured information across frames (temporal modeling). In order to leverage an image transformer model for video tasks, one key thing is how to perform temporal modeling. A simple baseline, termed space-only model, process each video frame independently by an image model. Given $\boldsymbol{x} \in \mathbb{R}^{T \times H \times W \times C}$, where $T$ is the number of frames, space-only model will get $T$ [class] tokens where each [class] token stands for the representation of each frame. These $T$ [class] tokens will be averaged as a way of temporal modeling for final prediction. In order to enhance the capability of temporal modeling, recent works (Bertasius et al., 2021; Arnab et al., 2021; Zhang et al., 2021b) introduce space-time model by adding new temporal modules to image models. These models are now the top performers on most video action recognition benchmarks, however, their training costs are prohibitively high due to full finetuning. Given the increasingly larger but more powerful pre-trained image models, in this work, we study how to efficiently adapt them for video action recognition.

### 3.2 Spatial Adaptation

Since image pre-trained models have been trained on large-scale datasets and demonstrated strong transferability to downstream tasks, we believe they could achieve good spatial modeling in video action recognition with minimal finetuning.

Inspired by efficient finetuning techniques (Houlsby et al., 2019; Lester et al., 2021; Li & Liang, 2021; Ben Zaken et al., 2022) in NLP, we adopt Adapter (Houlsby et al., 2019) due to its simplicity. As shown in Fig. 2(a), Adapter is a bottleneck architecture which consists of two fully connected (FC) layers and an activation layer in the middle. The first FC layer projects the input to a lower dimension and the second FC layer projects it back to the original dimension. To adapt the pre-trained spatial features to target video data, we add an Adapter after the self-attention layer as shown in Fig. 2(c), which we term as spatial adaptation. During training, all the other layers of the transformer model are frozen while only the Adapters are updated. In Table 1, we show that our spatial adaptation strategy achieves comparable performance with the full finetuned space-only baseline. This indicates that spatial adaptation helps the frozen image model to learn good spatial representations from video data. However, the overall performance after spatial adaptation still has a large gap to a full finetuned video model because spatial adaptation alone lacks the ability to learn temporal information in videos.

### 3.3 TEMPORAL ADAPTATION

To capture temporal information more effectively, previous methods usually incorporate new temporal modules to pre-trained image models because it is commonly believed that image models cannot infer temporal structured information in videos. However, adding new temporal modules, either temporal attention (Bertasius et al., 2021; Zhang et al., 2021b) or temporal encoder/decoder (Arnab et al., 2021; Lin et al., 2022), will introduce sizable number of extra tunable parameters. In addition, these new modules require full finetuning, which is inefficient.

To address this problem, we present a new strategy: *reuse the pre-trained self-attention layer in the image model to do temporal modeling*. More specifically, we denote the original self-attention layer as S-MSA for spatial modeling, and the reused self-atentnion layer as T-MSA for temporal modeling. As shown in Fig. 2(d), we put T-MSA in front of S-MSA. Now given the video patch embedding $\boldsymbol{z} \in \mathbb{R}^{T\times(N+1)\times D}$, we first reshape it into $\boldsymbol{z}^T \in \mathbb{R}^{(N+1)\times T\times D}$, where $N = HW/P^2$ is the number of spatial patches and $T$ is the number of frames. Then we feed $\boldsymbol{z}^T$ into the T-MSA where it tries to learn the relationship among the $T$ frames. Note that T-MSA and S-MSA are the same layer (i.e., pre-trained MSA in the image model) and kept frozen during model tuning, but just applied to different input dimensions. This explicit operation helps our model with enhanced temporal modeling, while keeping the number of parameters fixed. In the end, similar to spatial adaptation, we add another Adapter after the reused temporal attention layer to adapt its features on video data, which we term as temporal adaptation. The structure of the Adapter is the same as in spatial adaptation but without the skip connection. The reason is we want to initialize the adapted model to be close to the original model (Houlsby et al., 2019), thus we need to initialize the adapter to zero and remove the skip connection here to detach the effect of temporal adaptation at the beginning of training. As seen in Table 1, temporal adaptation helps to close the gap to full finetuned video models while only introducing another lightweight Adapter into the transformer block.

### 3.4 JOINT ADAPTATION

Spatial and temporal adaptation are performed sequentially to different input dimensions with their individual purposes. It would be desirable to jointly tune the representations for spatiotemporal reasoning. To this end, we further introduce an Adapter in parallel to the MLP layer, which we term as joint adaptation. This Adapter has the same structure as the one in temporal adaptation.

The final structure of a transformer block in our proposed AIM is presented in Fig. 2(e). The computation of the adapted block can be written as

$$\boldsymbol{z}_l^T = \boldsymbol{z}_{l-1} + \text{Adapter}(\text{T-MSA}(\text{LN}(\boldsymbol{z}_{l-1}))) \tag{3}$$

$$\boldsymbol{z}_l^S = \boldsymbol{z}_l^T + \text{Adapter}(\text{S-MSA}(\text{LN}(\boldsymbol{z}_l^T))) \tag{4}$$

$$\boldsymbol{z}_l = \boldsymbol{z}_l^S + \text{MLP}(\text{LN}(\boldsymbol{z}_l^S)) + s \cdot \text{Adapter}(\text{LN}(\boldsymbol{z}_l^S)) \tag{5}$$

where $\boldsymbol{z}_l^T$, $\boldsymbol{z}_l^S$, $\boldsymbol{z}_l$ denotes the temporal adapted, spatial adapted, and jointly adapted output in the $l$-th transformer block. Here, $s$ is a scaling factor to control the weight of the output from Adapter. For the final prediction, we simply take the average of the [class] tokens of each input frame and feed it to the classification head.

## 4 EXPERIMENTS

**Datasets.** We evaluate the proposed method on four widely adopted video action recognition benchmarks, Kinetics-400 (K400) (Kay et al., 2017), Kinetics-700 (K700) (Carreira et al., 2019), Something-something-v2 (SSv2) (Goyal et al., 2017) and Diving-48 (Li et al., 2018). K400 contains around 240K training videos and 20K validation videos in 400 human action classes. The videos are all trimmed to around 10 seconds. K700 is an extended version of K400 which contains around 530K training videos and 34K validation videos in 700 classes. SSv2 contains 168.9K training videos and 24.7K validation videos in 174 classes. SSv2 is more challenging because it requires stronger temporal modeling (Sevilla-Lara et al., 2021). Diving-48 contains 15.9K training videos and 2K validation videos in 48 fine-grained diving actions. It is designed to be unbiased towards static representations, which means a model cannot simply rely on the objects or background to determine the action.

Table 1: Effectiveness of proposed components. We compare to three baselines on Something-something-v2 dataset. Views = #frames × #temporal × #spatial.

| Methods | Pretrain | Param (M) | Tunable Param (M) | Top-1 | Top-5 | Views |
|---|---|---|---|---|---|---|
| Frozen space-only | IN-21K | 86 | 0.1 | 15.1 | 36.9 | 8×1×3 |
| Finetuned space-only | IN-21K | 86 | 86 | 36.2 | 68.1 | 8×1×3 |
| Finetuned space-time (Bertasius et al., 2021) | IN-21K | 121 | 121 | 59.5 | 85.6 | 8×1×3 |
| Frozen space-only + spatial adaptation | IN-21K | 89 | 3.7 | 36.7 | 68.3 | 8×1×3 |
| + temporal adaptation | IN-21K | 97 | 10.8 | 61.2 | 87.7 | 8×1×3 |
| + joint adaptation (AIM) | IN-21K | 100 | 14.3 | **62.0** | 87.9 | 8×1×3 |
| AIM | CLIP | 100 | 14.3 | **66.4** | 90.5 | 8×1×3 |

### 4.1 Effectiveness of Components

To demonstrate the effectiveness of our proposed components in Sec. 3, we compare our method to three baselines. The first baseline is a frozen space-only model. Recall in Sec. 3.1, space-only model processes input frames independently and performs temporal average pooling in the end. We freeze the image backbone and only tune the classification head, which is also known as linear probing (He et al., 2020). The second baseline is a full finetuned space-only model. It should be able to learn spatial information from video data, but still has difficulties in capturing temporal information. The third baseline is a full finetuned space-time video model, which should serves as oracle. Here we choose TimeSformer (Bertasius et al., 2021) because we are based on the same ViT-B backbone and share a similar structure (*i.e.*, divided space-time attention).

In the experiments, we use the ViT-B/16 pre-trained on IN-21K as image backbone, and we compare the proposed method with the baselines on SSv2 (Goyal et al., 2017) where temporal modeling is critical. The results for three baselines are shown in Tab. 1 top. We can see that the frozen space-only model only needs to tune 0.1M parameters, but it also performs much worse than the full finetuned video model (15.1% vs 59.5%). Full finetuning the space-only model allows it to learn improved spatial representations from video data and largely improves the performance (15.1% $\rightarrow$ 36.2%). However, it also significantly increases the number of tunable parameters and still has a large gap from the full finetuned video model due to lack of temporal modeling. The third baseline, full finetuned video model, achieves the highest accuracy due to its strong spatiotemporal reasoning capability, but the number of tunable parameters increases again to 121M.

Our goal is to add a few tunable parameters to the frozen space-only model and close the gap to full finetuned video model. As shown in Tab. 1 bottom, after spatial adaptation, the frozen space-only model achieves comparable performance with the full finetuned space-only model (36.7% vs 36.2%), with significantly less number of tunable parameters (3.7M vs 86M). This means spatial adaptation is able to help frozen image models to achieve good spatial modeling on video data. In addition, adding temporal adaptation further boosts the performance to 61.2%, which is even higher than the full finetuned video model. This indicates that our temporal adaptation introduces strong temporal modeling to the space-only model. Finally, joint adaptation is incorporated to tune the features for improved spatiotemporal reasoning, which is our method AIM. We not only close the gap to full finetuned space-time video model but obtain higher accuracy (62% vs 59.5%) with fewer number of tunable parameters (14.3M vs 86M). These results successfully validate the effectiveness of our proposed adaptation strategies.

Furthermore, our method could easily take advantage of stronger pre-trained image models and adapt them for video action recognition. For example, simply switch the ViT-B/16 pre-trained on IN-21K to CLIP pre-trained, we obtain another accuracy boost (62.0% $\rightarrow$ 66.4%)

### 4.2 Comparisons to the State of the art

In this section, we compare the proposed method with state-of-the-art video models on four video action recognition benchmarks. For all the experiments, we use the ViT models pre-trained by CLIP (Radford et al., 2021). We mostly follow the training settings in Liu et al. (2022), and more implementation details can be found in Appendix.

Table 2: Comparison to state-of-the-art on Kinetics-400. Views = #frames $\times$ #temporal $\times$ #spatial.

| Methods | Pretrain | GFLOPs | Param (M) | Tunable Param (M) | Top-1 | Top-5 | Views |
|---|---|---|---|---|---|---|---|
| MViT-B (Fan et al., 2021) | - | 4095 | 37 | 37 | 81.2 | 95.1 | 64×3×3 |
| UniFormer-B (Li et al., 2021) | IN-1K | 3108 | 50 | 50 | 83.0 | 95.4 | 32×4×3 |
| TimeSformer-L (Bertasius et al., 2021) | IN-21K | 7140 | 121 | 121 | 80.7 | 94.7 | 64×1×3 |
| ViViT-L/16×2 FE (Arnab et al., 2021) | IN-21K | 3980 | 311 | 311 | 80.6 | 92.7 | 32×1×1 |
| VideoSwin-L (Liu et al., 2022) | IN-21K | 7248 | 197 | 197 | 83.1 | 95.9 | 32×4×3 |
| MViTv2-L (312 ↑) (Li et al., 2022) | IN-21K | 42420 | 218 | 218 | 86.1 | 97.0 | 32×3×5 |
| MTV-L (Yan et al., 2022) | JFT | 18050 | 876 | 876 | 84.3 | 96.3 | 32×4×3 |
| TokenLearner-L/10 (Ryoo et al., 2021) | JFT | 48912 | 450 | 450 | 85.4 | 96.3 | 64×4×3 |
| PromptCLIP A7 (Ju et al., 2021) | CLIP | - | - | - | 76.8 | 93.5 | 16×5×1 |
| ActionCLIP (Wang et al., 2021a) | CLIP | 16890 | 142 | 142 | 83.8 | 97.1 | 32×10×3 |
| X-CLIP-L/14 (Ni et al., 2022) | CLIP | 7890 | 420 | 420 | 87.1 | 97.6 | 8×4×3 |
| EVL ViT-L/14 (Lin et al., 2022) | CLIP | 8088 | 368 | 59 | 87.3 | - | 32×3×1 |
| AIM ViT-B/16 | CLIP | 606 | 97 | 11 | 83.9 | 96.3 | 8×3×1 |
| AIM ViT-B/16 | CLIP | 1214 | 97 | 11 | 84.5 | 96.6 | 16×3×1 |
| AIM ViT-B/16 | CLIP | 2428 | 97 | 11 | 84.7 | 96.7 | 32×3×1 |
| AIM ViT-L/14 | CLIP | 2802 | 341 | 38 | 86.8 | 97.2 | 8×3×1 |
| AIM ViT-L/14 | CLIP | 5604 | 341 | 38 | 87.3 | 97.6 | 16×3×1 |
| AIM ViT-L/14 | CLIP | 11208 | 341 | 38 | **87.5** | 97.7 | 32×3×1 |

#### 4.2.1 Results on Kinetics-400 and Kinetics-700

Tab. 2 presents the comparisons with state-of-the-art video models on K400 dataset. First, we can see that with ViT-B/16 backbone, our method only needs to tune 11M parameters for competitive performance, which is much smaller than previous video models. Taking input of 8 frames as an example, AIM ViT-B/16 achieves 83.9% top-1 accuracy while only requiring 606 GFLOPs. When using 16 input frames, our method even outperforms MTV-L (Yan et al., 2022), which requires more than 10$\times$ computations (1214 vs 18050 GFLOPs). When switching to larger backbone ViT-L/14, we achieve the highest accuracy 87.5% on K400 dataset, with 38M tunable parameters.

Note that several works also leverage CLIP pre-trained models to do video action recognition. However, ActionCLIP (Wang et al., 2021a) and X-CLIP (Ni et al., 2022) are multimodal methods which require additional text branch and tune the whole model end-to-end. PromptCLIP (Ju et al., 2021) applies prompt tuning (Lester et al., 2021) to CLIP and adds several temporal blocks for temporal modeling. EVL (Lin et al., 2022) introduces a new decoder branch to learn temporal information. However, AIM simply re-uses image pre-trained self-attention for temporal modeling. This makes AIM much simpler than previous methods, yet achieving better performance at much less tunable parameters. The simplicity also makes AIM much easier to adapt to different model architectures (single modal or multi-modal models). But previous methods such as ActionCLIP/X-CLIP/PromptCLIP cannot leverage pure image backbone because they need and additional text branch.

Furthermore, we evaluate our method on K700 dataset in Tab. 4. We can see that AIM ViT-B/16 with 11M tunable parameters is able to outperform MTV-L (875M) and MViTv2-B (51M). This again justifies our efficient adaptation pipeline. Note that MViTv2-L (218M) achieves higher accuracy than ours, but they use larger input resolution (312 vs 224) and more input frames (40 vs 32).

#### 4.2.2 Results on Something-Something-v2

Tab. 3 presents the performance comparisons on SSv2. Based on CLIP ViT-L/14, our method achieves competitive or better performance than most prior arts. In terms of fair comparison to EVL, which also uses CLIP pre-trained image encoder, we achieve significantly higher accuracy ($70.6\% > 66.7\%$), while introducing less tunable parameters (50M $<$ 175M). Note that to introduce temporal modeling into image model, EVL adds 12 layers of decoder blocks, while our method simply reuse image pre-trained self-attention layers to achieve stronger temporal modeling .

However, our method falls behind some full finetuned video models (Girdhar et al., 2022; Li et al., 2022; 2021). One reason is that SSv2 is a "temporal-heavy" dataset (Sevilla-Lara et al., 2021), which requires model to really understand the temporal evolution within a video. In order to obtain high accuracy, most previous video models are first pre-trained on some video datasets (such as K400/K600) to learn good spatiotemporal representations, then finetuned on SSv2. But our method

Table 3: Comparison to state-of-the-art on Something-Something-v2. K400$^{\dagger}$/K600$^{\dagger}$ indicates the model is pre-trained on both IN-21K and K400/K600.

| Methods | Pretrain | GFLOPs | Param (M) | Tunable Param (M) | Top-1 | Top-5 | Views |
|---|---|---|---|---|---|---|---|
| TimeSformer-L (Bertasius et al., 2021) | IN-21K | 7140 | 121 | 121 | 62.4 | - | 64×1×3 |
| MTV-B (Yan et al., 2022) | IN-21K | 4790 | 310 | 310 | 67.6 | 90.4 | 32×4×3 |
| MViT-B (Fan et al., 2021) | K400 | 510 | 37 | 37 | 67.1 | 90.8 | 32×1×3 |
| MViTv2-B (Li et al., 2022) | K400 | 675 | 51 | 51 | 70.5 | 92.7 | 40×1×3 |
| ViViT-L/16×2 (Arnab et al., 2021) | K400$^{\dagger}$ | 11892 | 311 | 311 | 65.4 | 89.8 | 16×4×3 |
| VideoSwin-B (Liu et al., 2022) | K400$^{\dagger}$ | 963 | 89 | 89 | 69.6 | 92.7 | 32×1×1 |
| Omnivore (Girdhar et al., 2022) | K400$^{\dagger}$ | - | - | - | 71.4 | 93.5 | 32×1×3 |
| MViTv2-L (312 ↑) (Li et al., 2022) | K400$^{\dagger}$ | 8484 | 213 | 213 | **73.3** | 94.1 | 32×1×3 |
| UniFomer-B (Li et al., 2021) | K600$^{\dagger}$ | 777 | 50 | 50 | 71.2 | 92.8 | 32×1×3 |
| CoVeR (Zhang et al., 2021a) | JFT-3B | - | - | - | 70.9 | - | - |
| EVL ViT-B/16 (Lin et al., 2022) | CLIP | 2047 | 182 | 86 | 62.4 | - | 32×1×3 |
| EVL ViT-L/14 Lin et al. (2022) | CLIP | 9641 | 484 | 175 | 66.7 | - | 32×1×3 |
| AIM ViT-B/16 | CLIP | 624 | 100 | 14 | 66.4 | 90.5 | 8×1×3 |
| AIM ViT-B/16 | CLIP | 1248 | 100 | 14 | 68.1 | 91.8 | 16×1×3 |
| AIM ViT-B/16 | CLIP | 2496 | 100 | 14 | 69.1 | 92.2 | 32×1×3 |
| AIM ViT-L/14 | CLIP | 2877 | 354 | 50 | 67.6 | 91.6 | 8×1×3 |
| AIM ViT-L/14 | CLIP | 5754 | 354 | 50 | 69.4 | 92.3 | 16×1×3 |
| AIM ViT-L/14 | CLIP | 11508 | 354 | 50 | 70.6 | 92.7 | 32×1×3 |

Table 4: Comparisons on Kinetics-700.

| Method | Pretrain | Tunable Param | Top-1 |
|---|---|---|---|
| VidTR-L (Zhang et al., 2021b) | IN-21K | 91 | 70.2 |
| MTV-L (Yan et al., 2022) | IN-21K | 876 | 75.2 |
| MViTv2-B (Li et al., 2022) | - | 51 | 76.6 |
| MViTv2-L (312 ↑) (Li et al., 2022) | IN-21K | 218 | 79.4 |
| AIM ViT-B/16 | CLIP | 11 | **76.9** |

Table 5: Comparisons on Diving-48.

| Method | Pretrain | Tunable Param | Top-1 |
|---|---|---|---|
| TimeSformer-L (Bertasius et al., 2021) | IN-21K | 121 | 81.0 |
| VideoSwin-B (Liu et al., 2022) | IN-21K | 88 | 81.9 |
| SIFAR-B-14 (Fan et al., 2022) | IN-21K | 87 | 87.3 |
| ORViT (Herzig et al., 2022) | IN-21K | 160 | 88.0 |
| AIM ViT-B/16 | CLIP | 11 | **88.9** |

still starts from the image pre-trained model. Another reason is that simply reusing the image pre-trained self-attention for temporal modeling may not be able to fully capture the complicated temporal information in SSv2 videos. This suggests that we need to conduct more temporal adaptation for these challenging "temporal-heavy" datasets.

### 4.2.3 Results on Diving-48

A diving class in Diving-48 (Li et al., 2018) is defined by the combination of takeoff, movements in flight and entry, thus it requires the model to differentiate such fine-grained actions. As shown in Tab. 5, our method with 11M tunable parameters outperforms all prior methods. Comparing to ORViT (Herzig et al., 2022), despite they leverage additional object tracking model, our method still outperforms it with much less tunable parameters. This suggests that efficient finetuning can handle fine-grained action recognition.

## 5 Discussion

**Different Pre-trained Models.** Here we demonstrate the effectiveness of AIM on different pre-trained models. In Table 6, we first show AIM based on ViT-B backbone. We compare AIM to TimeSformer because we use the same backbone (ViT-B) and have a similar structure (i.e., both using divided space-time attention). As can be seen, AIM achieves better performance than full finetuned TimeSformer under both IN-21K and CLIP pre-trained weights. Then we apply AIM to Swin-B backbone and compare it to VideoSwin when we both use Swin-B and IN-21K pre-training. Similarly, AIM achieves comparable performance with full finetuned VideoSwin.

Table 6: Performance of using different pre-trained models on K400.

| Model | Backbone | Pretrain | Tunable Param (M) | Mem (G) | Time (H) | Top-1 |
|---|---|---|---|---|---|---|
| TimeSformer | ViT-B | IN-21K | 121 | 10 | 20 | 78.5 |
| AIM | ViT-B | IN-21K | 11 | 7 | 15 | 78.8 |
| TimeSformer | ViT-B | CLIP | 121 | 10 | 20 | 82.0 |
| AIM | ViT-B | CLIP | 11 | 7 | 15 | 83.9 |
| VideoSwin-B | Swin-B | IN-21K | 88 | 18 | 64 | 82.7 |
| AIM | Swin-B | IN-21K | 9.2 | 9 | 37 | 82.1 |

**Data Efficiency.** One advantage of our efficient tuning paradigm is that we can keep the well pre-trained image representations intact. In the scenario where downtream data is insufficient, our

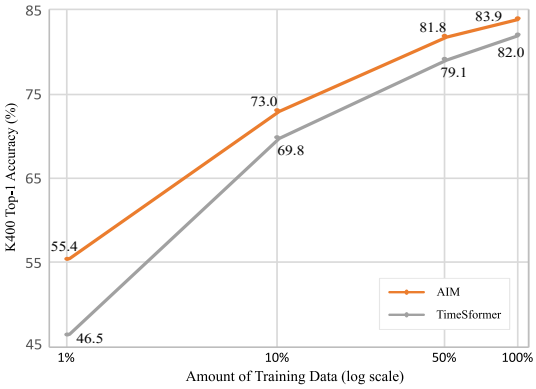


Figure 3: Data efficiency comparison. AIM outperforms full finetuned TimeSformer under all scenarios, especially in low data regime.

Table 7: Effect of position of Adapters. Skip means adding Adapters every two blocks.

| Position | Tunable Param (M) | Top-1 |
|---|---|---|
| Bottom 6 | 5.6 | 80.7 |
| Top 6 | 5.6 | 83.3 |
| Skip | 5.6 | 83.2 |
| All | 11 | **83.9** |

Table 8: Effect of bottleneck ratio of Adapters.

| Ratio | Tunable Param (M) | Top-1 |
|---|---|---|
| 0.0625 | 3 | 83.3 |
| 0.125 | 5.6 | 83.4 |
| 0.25 | 11 | **83.9** |
| 0.5 | 21 | 83.8 |

method will be less prone to over-fitting problem compared to full finetuning. In Fig. 3, we compare AIM with full finetuned TimeSformer under different amounts of training data on K400. For fair comparison, both AIM and TimeSformer use CLIP pre-trained ViT-B/16 as backbone. We can observe that under all scenarios, our method AIM outperforms full finetuned TimeSformer. In particular, when the amount of data becomes less, the advantage of AIM becomes larger. For example, when there is only $1\%$ of training data, we outperform TimeSformer by a significant margin of $8.9\%$.

**Training Cost.** Tab. 6 also shows the training time (hours) and memory cost (GB) of our method and full finetuning on different backbones. All metrics are measured on 8 Tesla V100 GPUs. Compared to TimeSformer, we reduce the memory cost by 30% and training time by 25%. Compared to VideoSwin, we reduce the memory cost by 50% and training time by 42%.

**Position of Adapters.** By default, we add Adapters to every ViT block (12 blocks in total). Here we study the effect of adding Adapters in different layers. We add Adapters to the bottom 6 blocks (close to the input), top 6 blocks (close to the output) and one every two blocks. All these variants have the same number of tunable parameters. As can be seen in Tab. 7, adding Adapters to the bottom 6 blocks yields much worse performance than others. We hypothesize that the shallow layers learn generic representations which do not need much adaptation, while deeper layers learn task-specific features like temporal information thus feature adaptation is important. Adding Adapters to the top 6 blocks achieves comparable performance with adding to all blocks while saving half of the parameters. This could serve as a good candidate when training resources are more limited.

**Bottleneck Ratio of Adapters.** By tuning the bottleneck ratio of Adapters, we can easily control the number of tunable parameters. Here we study how the bottleneck ratio of Adapters affects the final performance. The results in Tab. 8 reveal that a larger bottleneck ratio tends to achieve better performance, but it will also introduce more tunable parameters. The performance plateaus after bottleneck ratio goes beyond 0.25. Note that a small ratio of 0.0625 could still achieve 83.3% top-1 accuracy on K400, which is competitive among state-of-the-art video models in Tab. 2 while introducing only 3M tunable parameters.

## 6 Conclusion

In this work, we propose a new way to efficiently transfer pre-trained image models for video action recognition. By using Adapters, we introduce spatial adaptation, temporal adaptation and joint adaptation to gradually add spatiotemporal reasoning to an image model. Since most of the model parameters are kept frozen and only newly added Adapters are updated, our training cost is substantially lower than other full finetuned video models. Yet we achieve comparable or even better performance than prior arts on four benchmarks. Our method is simple and generally applicable, which has the potential to leverage more powerful image foundation models in the future. Despite all the benefits, one limitation of our work is that our simple strategy of reusing spatial attention for temporal modeling. It might not be strong enough for challenging video datasets that focus more on temporal dimension. Since video temporal modeling can be viewed as a form of sequence modeling, we might be able to reuse pre-trained weights from text or audio models instead of image models. We leave these explorations as future work.

## REFERENCES


Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6836–6846, 2021.

Hyojin Bahng, Ali Jahanian, Swami Sankaranarayanan, and Phillip Isola. Visual prompting: Modifying pixel space to adapt pre-trained models. *arXiv preprint arXiv:2203.17274*, 2022.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT Pre-Training of Image Transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.1. URL `https://aclanthology.org/2022.acl-short.1`.

Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, volume 2, pp. 4, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *arXiv preprint arXiv:2205.13535*, 2022.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9640–9649, 2021.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12124–12134, 2022.

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 *International Conference on Learning Representations*, 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.

Quanfu Fan, Chun-Fu Chen, and Rameswar Panda. Can an image classifier suffice for action recognition? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=qhkFX-HLuHV.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6202–6211, 2019.

Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. *arXiv preprint arXiv:2205.09113*, 2022.

Rohit Girdhar, Mannat Singh, Nikhila Ravi, Laurens van der Maaten, Armand Joulin, and Ishan Misra. Omnivore: A single model for many visual modalities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16102–16112, 2022.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pp. 5842–5850, 2017.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=0RDcd5Axok.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022b.

Roei Herzig, Elad Ben-Avraham, Karttikeya Mangalam, Amir Bar, Gal Chechik, Anna Rohrbach, Trevor Darrell, and Amir Globerson. Object-region video transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3148–3159, 2022.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pp. 4904–4916. PMLR, 2021.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. *arXiv preprint arXiv:2203.12119*, 2022.

Shibo Jie and Zhi-Hong Deng. Convolutional bypasses are better vision transformer adapters. *arXiv preprint arXiv:2207.07039*, 2022.

Chen Ju, Tengda Han, Kunhao Zheng, Ya Zhang, and Weidi Xie. Prompting visual-language models for efficient video understanding. *arXiv preprint arXiv:2112.04478*, 2021.

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.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Kunchang Li, Yali Wang, Gao Peng, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unified transformer for efficient spatial-temporal representation learning. In *International Conference on Learning Representations*, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353.

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 *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4794–4804, 2022. doi: 10.1109/CVPR52688.2022.00476.

Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 513–528, 2018.

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

Ziyi Lin, Shijie Geng, Renrui Zhang, Peng Gao, Gerard de Melo, Xiaogang Wang, Jifeng Dai, Yu Qiao, and Hongsheng Li. Frozen clip models are efficient video learners. *arXiv preprint arXiv:2208.03550*, 2022.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 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*, pp. 3202–3211, 2022.

Bolin Ni, Houwen Peng, Minghao Chen, Songyang Zhang, Gaofeng Meng, Jianlong Fu, Shiming Xiang, and Haibin Ling. Expanding language-image pretrained models for general video recognition. *arXiv preprint arXiv:2208.02816*, 2022.

Zhiwu Qing, Shiwei Zhang, Ziyuan Huang, Xiang Wang, Yuehuan Wang, Yiliang Lv, Changxin Gao, and Nong Sang. Mar: Masked autoencoders for efficient action recognition. *arXiv preprint arXiv:2207.11660*, 2022.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Michael S Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: Adaptive space-time tokenization for videos. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=z-l1kpDXs88.

Laura Sevilla-Lara, Shengxin Zha, Zhicheng Yan, Vedanuj Goswami, Matt Feiszli, and Lorenzo Torresani. Only time can tell: Discovering temporal data for temporal modeling. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 535–544, 2021.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205, 2021.

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.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *arXiv preprint arXiv:2203.12602*, 2022.

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

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

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 568–578, 2021b.

Wenhui Wang et al. Image as a Foreign Language: BEiT Pretraining for All Vision and Vision-Language Tasks. *arXiv preprint arXiv:2208.10442*, 2022.

Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14668–14678, 2022.

Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9653–9663, 2022.

Shen Yan, Xuehan Xiong, Anurag Arnab, Zhichao Lu, Mi Zhang, Chen Sun, and Cordelia Schmid. Multiview transformers for video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3333–3343, 2022.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. Differentially private fine-tuning of language models. In *International Conference on Learning Representations*, 2021.

Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 558–567, 2021a.

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*, 2021b.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling Vision Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

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*, 2021a.

Yanyi Zhang, Xinyu Li, Chunhui Liu, Bing Shuai, Yi Zhu, Biagio Brattoli, Hao Chen, Ivan Marsic, and Joseph Tighe. Vidtr: Video transformer without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13577–13587, 2021b.

Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. Reduce communication costs and preserve privacy: Prompt tuning method in federated learning. *arXiv preprint arXiv:2208.12268*, 2022.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 13001–13008, 2020.

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2021.

# A Implementation details

## A.1 Kinetics 400/700

We add spatial/temporal/joint adaptation in every ViT block as shown in Fig. 2. The bottleneck ratios of all adapters are set to 0.25 and the scaling factor is set to 0.5. The first FC layer in Adapters is randomly initialized and the second FC layer is initialized to zero. In this way, the adapted model is close to the pre-trained model at the beginning of training. We largely follow the training settings and data augmentations in Liu et al. (2022). Specifically, the model is trained for 30 epochs using AdamW (Kingma & Ba, 2014) optimizer with a batchsize of 64. The base learning rate is 3e-4 and weight decay is 5e-2. The learning rate is warmed up from 0 in the first 3 epochs and then decays following a cosine schedule. The stochastic depth rate is 0.2 for both ViT-B and ViT-L. For inference, we sample three clips along the temporal dimension. The final performance is evaluated by the ensemble of three views. We evaluate the model on 8, 16, 32 frames and the sampling interval is 16, 8, 4, respectively.

## A.2 Something-something-v2

We add spatial/temporal/joint adaptation in every ViT block as shown in Fig. 2. We additionally add one adapter before T-MSA to enhance the temporal modeling. The bottleneck ratios of all adapters are set to 0.25 and the scaling factor is set to 0.5. We follow Liu et al. (2022) to use stronger data augmentations including label smoothing, RandAugment (Cubuk et al., 2020) and random erasing (Zhong et al., 2020). The model is trained for 50 epochs using AdamW (Kingma & Ba, 2014) optimizer. The other training settings are the same as Kinetics-400. We uniformly sample 8, 16, 32 frames in the experiments. For inference, we sample three spatial crops. The final performance is evaluated by the ensemble of three views.

## A.3 Diving-48

We add spatial/temporal/joint adaptation in every ViT block as shown in Fig. 2. The bottleneck ratios of all adapters are set to 0.25 and the scaling factor is set to 0.5. The model is trained for 50 epochs. The other training settings and data augmentations are the same as K400. We uniformly sample 8, 16, 32 frames in the experiments. For inference, we only sample 1 temporal clip.

# B More comparisons of training cost

In Table 6, we demonstrate the training efficiency of AIM based on ViT-B and Swin-B backbones. In this section, we show more comparisons with full finetuned baselines based on ViT-L and Swin-L backbones. The results are shown in Table 9. We can see that TimeSformer with a ViT-L backbone needs 21.2G GPU memory, and VideSwin with a Swin-L backbone cannot fit into an 8 Tesla V100 32G GPU server. In both cases, AIM can significantly reduce the memory usage to 14.3G and 13.7G, respectively. This makes large model training more memory-friendly (runnable on most GPUs with 15G memory and more) , and thus more affordable for most researchers and practioners.

Furthermore, beyond memory saving, optimizing number of tunable parameters has potential benefits in other applications such as communication-efficient distributed learning (e.g., federated learning where the tunable model parameters are communicated between the central server and local

Table 9: Comparisons of training memory with large full finetuned baselines.

| Model | Backbone | Mem (G) |
|---|---|---|
| TimeSformer Bertasius et al. (2021) | ViT-L | 21.2 |
| AIM | ViT-L | 14.3 |
| VideoSwin Liu et al. (2022) | Swin-L | Out of Memory |
| AIM | Swin-L | 13.7 |

Table 10: Comparisons with EVL under different pre-trained datasets.

| Model | Backbone | Pretrain | Tunable Param (M) | Mem (G) | Time (H) | Top-1 |
|---|---|---|---|---|---|---|
| EVL Lin et al. (2022) | ViT-B | IN-21K | 36.3 | 4.2 | 29 | 75.4 |
| AIM | ViT-B | IN-21K | 11 | 7 | 15 | 78.8 |
| EVL Lin et al. (2022) | ViT-B | CLIP | 36.3 | 4.2 | 29 | 82.9 |
| AIM | ViT-B | CLIP | 11 | 7 | 15 | 83.9 |

clients) and privacy preserving federated learning Zhao et al. (2022); Yu et al. (2021). Tuning less parameters could also be beneficial when the downstream data is limited because fully finetuning a large model on limited data may suffer from serious overfitting. This can be observed from the results in Fig 3 where AIM obtains larger accuracy improvements over the full finetuned baseline when there is only small amount of training data.

## C Comparison to EVL under different pre-trained datasets

In this section, we compare AIM with EVL Lin et al. (2022), which is the most recent SoTA image-to-video efficient finetuning method based on frozen pre-trained ViT. As shown in the Table 10, AIM consistently outperforms EVL under both IN-21K and CLIP pre-training as well.

## D Pseudo-code of the adapted ViT block

We show the PyTorch style pseudo-code in Algorithm 1

**Algorithm 1** Pseudo-code of an adapted ViT block

```
class TransformerBlock():

    def __init__(self):
        ## Layers in the original ViT block
        self.attn = MultiheadAttention(dim, num_head)
        self.norm1 = LayerNorm(dim)
        self.mlp = MLP(dim, mlp_ratio)
        self.norm2 = LayerNorm(dim)

        ## Adapters
        self.s_adapter = Adapter(dim)
        self.t_adapter = Adapter(dim)
        self.mlp_adapter = Adapter(dim)
        self.scale = scale

    def forward(x):
        ## x in shape [N+1, BT, D]

        ## temporal adaptation
        xt = rearrange(x, 'n (b t) d -> t (b n) d', t=num_frames)
        xt = self.t_adapter(self.attn(self.norm1(x)))
        xt = rearrange(x, 't (b n) d -> n (b t) d', n=num_patches)
        x = x + xt

        ## spatial adaptation
        x = x + self.s_adapter(self.attn(self.norm1(x)))

        ## joint adaptation
        x_norm = self.norm2(x)
        x = x + self.mlp(x_norm) + self.scale * self.mlp_adapter(x_norm)

        return x
```