AdaptFormer: Adapting Vision Transformers for Scalable Visual Recognition

Shoufa Chen[1*] **Chongjian Ge**[1*] **Zhan Tong**[2] **Jiangliu Wang**[2]
Yibing Song[2] **Jue Wang**[2] **Ping Luo**[1]
[1]The University of Hong Kong [2]Tencent AI Lab

Abstract

Pretraining Vision Transformers (ViTs) has achieved great success in visual recognition. A following scenario is to adapt a ViT to various image and video recognition tasks. The adaptation is challenging because of heavy computation and memory storage. Each model needs an independent and complete finetuning process to adapt to different tasks, which limits its transferability to different visual domains. To address this challenge, we propose an effective adaptation approach for Transformer, namely AdaptFormer, which can adapt the pre-trained ViTs into many different image and video tasks efficiently. It possesses several benefits more appealing than prior arts. Firstly, AdaptFormer introduces lightweight modules that only add less than 2% extra parameters to a ViT, while it is able to increase the ViT's transferability without updating its original pre-trained parameters, significantly outperforming the existing 100% fully fine-tuned models on action recognition benchmarks. Secondly, it can be plug-and-play in different Transformers and scalable to many visual tasks. Thirdly, extensive experiments on five image and video datasets show that AdaptFormer largely improves ViTs in the target domains. For example, when updating just 1.5% extra parameters, it achieves about 10% and 19% relative improvement compared to the fully fine-tuned models on Something-Something v2 and HMDB51, respectively. Code is available at https://github.com/ShoufaChen/AdaptFormer.

在每个不同的任务上都要微调，微调工作量大，就不能很好的迁移任务

参数量小但是效果却很好

适配器：即插即用，扩展性很好

高效

1 Introduction

There is a growing interest in adopting a general neural model to tackle a large variety of different tasks since it benefits in reducing the need for task-specific model design and training. Recently, Transformer [81] demonstrates great potential in this goal considering its success in various fields, *e.g.*, natural language processing (NLP) [27, 10, 82, 88], visual recognition [31, 79, 90, 63], dense prediction [83, 11, 98, 96, 86], Generative Adversarial Network (GAN) [52, 48], reinforcement learning (RL) [18, 16, 87], robotics [50, 25], and etc. However, existing literature in computer vision tend to focus on the *same network with task-specific weights* scenario, where a single network is used to train from scratch or fully fine-tune on a specific dataset, making it infeasible to maintain a separate model weight for every dataset when the number of task grows, especially for the increasing model capacity of state-of-the-art models (*e.g.*, ViT-G/14 [93] with over 1.8 billion parameters).

通用模型的流行

从后训练or全量微调 不可行

Different from prior arts, we step into the direction of developing *same network with almost same weights* and achieve superior performance than the full-tuning approach by only tuning less than 2% parameters, with the remaining over 98% parameters shared across different tasks. There are two challenges to learning universal representations using a single model. The first one lies in the pre-training stage, which requires algorithms that can learn well-generalized representations that are

*Equal contribution.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

easy to be applied to many tasks. Recent arts in self-supervised learning [12, 5, 43, 97, 85, 78, 35] can serve as a solution to this challenge. The second one, which is our main concern in this work, is to build an effective pipeline that can adapt the model obtained at the pre-training stage to various downstream tasks by tuning parameters as less as possible and keeping the left parameters frozen.

While fine-tuning pre-trained models has been widely studied in NLP [6, 46, 69, 70, 58, 56, 47, 92, 62, 42], this topic is seldomly explored in the vision, where full tine-tuning of model parameters is still the dominant strategy for adapting vision transformers. However, the full fine-tuning cannot satisfy the goal of *universal representation* as it assigns an independent set of weights for every task. Linear probing is a straightforward approach to maintaining the pre-trained model fixed by only tuning a specific lightweight classification head for every task. However, linear probing tends to have an unsatisfactory performance and misses the opportunity of pursuing strong but non-linear features [43], which indeed benefit deep learning. More recently, Bahng *et.al.,* [4] aimed to adapt pre-trained models by modifying raw input pixel space. Jia *et.al.,* [51] proposed Visual Prompt Tuning (VPT) to adapt transformer models for downstream vision tasks, which prepends several learnable parameters (*prompts*) to the patch embeddings and freezes the whole pre-trained backbone.

网络结构

Scaling factor的意义

In this work, we propose a lightweight module, namely AdaptFormer, to adapt vision transformers by updating the weights of AdaptFormer. We introduce learnable parameters from the model perspective, which is different from VPT, which inserts learnable parameters into the token space. Our AdaptFormer is conceptually simple yet effective. It consists of two fully connected layers, a non-linear activation function, and a scaling factor. This module is set in parallel to the feed-forward network (FFN) of the original ViT model, as shown in Figure 2b. This design is turned out to be effective for model transfer when processing scalable visual tokens for both image and video data (i.e., image data consists of a small scale of visual tokens while video data consists of a large scale). As shown in Figure 1, compared with the full-tuning strategy, AdaptFormer achieves comparable performance on video recognition with only about 0.1% tunable parameters. Meanwhile, with less than 2% tunable parameters, AdaptFormer surpasses the full-tuning solution by about 10% on top-1 accuracy. Similar approaches are also proposed in fine-tuning pre-trained language models (PLMs) [6, 46, 70, 42].

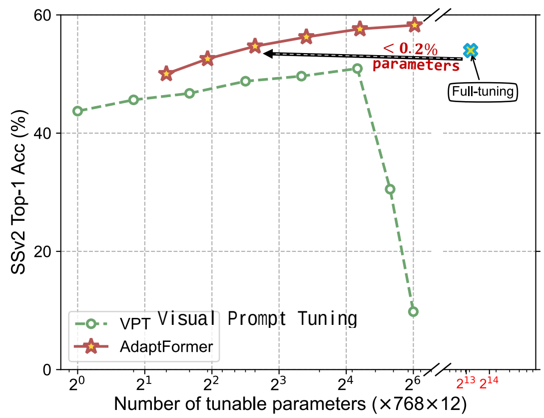


Figure 1: **Parameter-Accuracy trade-off.** We leverage ViT-Base as backbone and report top-1 accuracy on SSv2 dataset. AdaptFormer can surpass full-tuning with only 0.2% tunable parameters. More detailed results are shown in Table 1.

主要贡献

The key **contributions** of this paper are summarized as follows: **(1)** We propose a simple yet effective framework, namely AdaptFormer, for adapting vision transformers to a large variety of downstream visual recognition tasks and avoiding catastrophic interference with each other. To the best of our knowledge, this is the first work that explores efficient fine-tuning in video action recognition. **(2)** We ablate many design choices and demonstrate the superior robustness of AdaptFormer when parameters scale up. **(3)** Extensive experiments on various downstream tasks demonstrate that AdaptFormer outperforms existing fine-tuning approaches significantly. By demonstrating the effectiveness of AdaptFormer on multiple visual benchmarks, we hope our work could inspire the research communities to rethink the fine-tuning mechanism in computer vision and make progress toward a flexible yet universal Transformer model for visual recognition.

2 Related Works

In the proposed AdaptFormer, we mainly introduce a plug-and-play module for efficiently fine-tuning the current vision Transformer models. In this section, we perform a literature review on related works from two perspectives, *i.e.*, the vision Transformers, and efficient transfer learning for vision Transformers.

2.1 Transformer in Vision

The Transformer architecture is first introduced in [81] and has re-energized the natural language processing (NLP) field from then on [27, 10]. Inspired by its huge success, researches in the computer vision filed have also evolved into Transformer era since ViTs [31]. The strong capability of modeling long-range relation has facilitated Transformer in various vision tasks, including image classification [31, 63, 60], object detection [11, 98, 22], semantic/instance segmentation [86], video understanding [8, 2, 33, 57], point cloud modeling [95, 41], 3D Object Recognition [20] and even low-level processing [17, 59, 84]. Furthermore, transformers have advanced the vision recognition performance by a large-scale pretraining [21, 67, 13, 36, 43, 78, 71]. In such a situation, given the pre-trained Transformer models, which are more larger than the previously prevalent CNN backbones, one open question is how to fine-tune the big vision models so that they can be adapted into more specific down-stream tasks. To solve the open question, we propose AdaptFormer to transfer ViTs from the pre-trained pre-texts into the target tasks in a more effective and efficient way.

2.2 Efficient Transfer learning for Transformers

Transfer learning targets re-adopting a pre-trained model (either via the supervised or the unsupervised manner) as the starting point and further fine-tuning the specific model on a new task. In the NLP field, transferring the large pre-trained language models (PLMs) [27, 10] into downstream tasks has been the popular paradigm for a long time. Conventional arts [27, 10] set all the network parameters as learnable ones and adapt them to the target tasks. However, with the growth of model sizes and the complexity of the specific tasks, the conventional paradigm is inevitably limited by the huge computational burden. The NLP community has explored several ways for parameter-efficient transfer learning that only set a few parameters learnable and fine-tune them for efficiency. The pioneer works could be mainly categorized from the token [58, 56] and network perspectives [46, 47, 92, 40]. Basically speaking, the token-related methods [56, 58] typically prepend several learnable prefix vectors/tokens to the projected tokens within the multi-head self-attention layers (MHSA [81]). The philosophy behind it is to assist the pre-trained models in understanding downstream tasks with the guidance of extra token information. On the other hand, network-related methods [46, 47] integrate shallow modules to improve the model transferability. The introduced modules adapt the produced representations into the downstream tasks via features fusion.

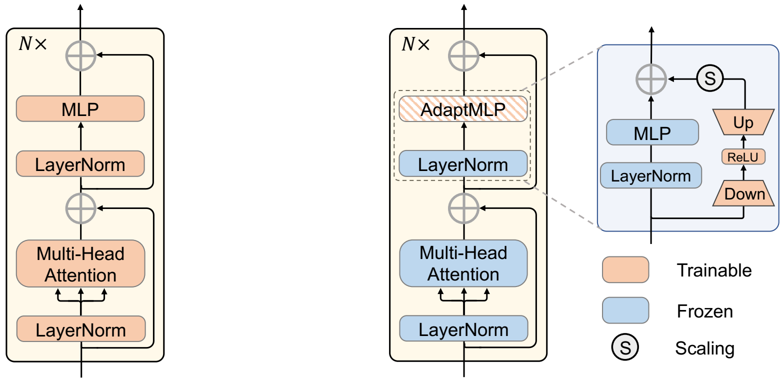
Recently, with the emergence of a much more large-scale dataset [26, 72, 74, 66, 53], increasing researchers in computer vision have adopted the homologous paradigm, *i.e.*, first pre-training and then fine-tuning, to advance the vision tasks. As for the second stage, traditional methods typically adopt the full-tuning arts in the downstream tasks. Rare attention has been drawn to the field of efficient adaptation, especially in the field of vision Transformers. Inspired by Prompting in NLP, [51] introduced the learnable tokens in exploring the efficient adaptation for ViTs. We empirically found that the performance of prompting is hindered by the scale of tokens. That is to say, for the tasks where the number of tokens is on a small scale, *e.g.*, image classification, Prompting is efficient for improving the model transferability. However, for larger scale tokens, *e.g.*, video understanding, Prompting presents limited potential. This observation motivates us to introduce AdaptFormer, which is effective in the scenarios of scalable visual tokens.

3 Approach

We propose AdaptFormer for efficiently transferring large pre-trained vision transformer models to downstream tasks, in both image and video domains. AdaptFormer attains strong transfer learning abilities by only fine-tuning a small number of extra parameters, circumventing catastrophic interference among tasks. We illustrate the overall framework of AdaptFormer in Figure 2b.

3.1 Preliminary and Notation

Vision Transformers (ViTs) are first introduced by [31] into vision recognition. A vanilla vision Transformer basically consists of a patch embedding layer and several consecutively connected encoders, as depicted in Figure 2a. Given an image $x \in \mathbb{R}^{H\times W\times 3}$, the patch embedding layer first splits and flatten the sample x into sequential patches $x_p \in \mathbb{R}^{N\times (P^2 d)}$, where (H, W) represents the *height* and *width* of the input image, (P, P) is the resolution of each image patch, d denotes



(a) Full fine-tuning. (b) AdaptFormer fine-tuning

Figure 2: **Comparison of previous *full* and our *AdaptFormer* fine-tuning.** AdaptFormer is conceptually simple by replacing the original MLP block with AdaptMLP, which consists of two branches, including the frozen branch (left) and the trainable `down` $\rightarrow$ `up` bottleneck module (right).

the output channel, and $N = HW/P^2$ is the number of image tokens. The overall combination of a prepended `[CLS]` token and the image tokens x_p are further fed into Transformer encoders for attention calculation.

Each Transformer encoder mainly consists of two types of sub-layers, *i.e.*, a multi-head self-attention layer (MHSA) and a MLP layer. In MHSA, the tokens are linearly projected and further re-formulated into three vectors, namely $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$. The self-attention calculation is performed on $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ by:

$$x'_\ell = \text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}})\boldsymbol{V}, \tag{1}$$

where x'_ℓ are the tokens produced by MHSA at the ℓ-th layer. The output tokens x'_ℓ are further sent to a LayerNorm [3] and a MLP block which is consisted of two fully connected layers with a GELU activation [45] in between. This process is formally formulated as follows,

$$x_\ell = \text{MLP}(\text{LN}(x'_\ell)) + x'_\ell, \tag{2}$$

where x_ℓ is the output of the ℓ-th encoder block. At the last transformer layer, the `[CLS]` is utilized for the final object recognition. We refer the readers to find more details in [31]. In our work, we replace the MLP layer with our AdaptMLP module for efficient fine-tuning purposes.

3.2 AdaptFormer

We propose a plug-and-play bottleneck module, namely AdaptMLP[2]. We denote the vision Transformer equipped with AdaptMLP as AdaptFormer.

Architecture. The design principle of AdaptFormer is simple yet effective, which is illustrated in Figure 2b. Compared to the vanilla full fine-tuning regime, AdaptFormer replaces the MLP block in the transformer encoder with *AdaptMLP*, which is consisted of two sub-branches. The MLP layer in the left branch is identical to the original network, while the right branch is an additionally introduced lightweight module for task-specific fine-tuning. Specifically, the right branch is designed to be a bottleneck structure for limiting the number of parameters purpose, which includes a down-projection layer with parameters $\boldsymbol{W}_{\text{down}} \in \mathbb{R}^{d \times \hat{d}}$, an up-projection layer with parameters $\boldsymbol{W}_{\text{up}} \in \mathbb{R}^{\hat{d} \times d}$, where $\hat{d}$ is the bottleneck middle dimension and satisfies $\hat{d} \ll d$. In addition, there is a ReLU layer [1] between these projection layers for non-linear property. This bottleneck module is connected to the original MLP network (left branch) through the residual connection via a scale factor s. For a specific input feature x'_ℓ, the right branch in AdaptMLP produces the adapted features, $\tilde{x}_\ell$, formally via:

$$\tilde{x}_\ell = \text{ReLU}(\text{LN}(x'_\ell) \cdot \boldsymbol{W}_{\text{down}}) \cdot \boldsymbol{W}_{\text{up}}. \tag{3}$$

[2]In this paper, we use the term 'AdaptMLP' to denote the designed module and the term 'AdaptFormer' to represent the fine-tuning framework for Vision Transformers. Unless otherwise specified, we apply AdaptFormer to fine-tune the vanilla ViT backbone [31] in this paper.

Then both the features $\tilde{x}_\ell$ and x'_ℓ are fused with x_ℓ by residual connection,

$$x_\ell = \mathrm{MLP}(\mathrm{LN}(x'_\ell)) + s \cdot \tilde{x}_\ell + x'_\ell. \tag{4}$$

Fine-tuning. During the fine-tuning phase, we only choose the newly added parameters to optimize and keep rest ones fixed. Specifically, the original model parts (blue blocks in Figure 2b) load weights from the pre-trained checkpoint and keeps parameters *frozen*. The newly added parameters (orange blocks) are updated on the specific data domain with the task-specific losses.

Inference. After fine-tuning, we still keep the shared parameters frozen as in the previous fine-tuning state, and additionally load the weights of the extra parameters that were fine-tuned in the previous stage. The single overall model is able to be adapted to multiple tasks with the assistance of lightweight introduced modules. 在标准模型上添加拥有不同权重参数的adapter，快速迁移任务

3.3 Discussion

Tunable parameters analysis. Our AdaptMLP module is lightweight. The total number of parameters introduced to per layer is $2 \times d \times \hat{d} + \hat{d} + d$, which includes biases parameters. The middle dimension $\hat{d}$ is a small value compared with d (AdaptFormer still obtains a decent performance even when $\hat{d} = 1$, as discussed in Sec. 4.5). Since most of the shared parameters are fixed and the number of newly introduced parameters is small ($< 2\%$ of the pre-trained model parameters), the total model size grows slowly when more downstream tasks are added.

Applicability. We note that AdaptMLP is a plug-and-play module that can be adaptively inserted into existing popular vision transformer architectures [31, 63, 83, 90, 23, 29] since all of the backbones share the same MLP layers even though they differ in the MHSA architectures (as shown in Figure 2b). Compared to our methods, we notice that recent prompt-related approaches insert trainable parameters into the token space, as illustrated in Figure 3. They prepend learnable parameters either into the embedded tokens before linear projection [58] or the key and value tokens after linear projection [51]. *Therefore, the prompt-related method can not be straightforwardly adapted to special MHSA variants, especially for the one that takes the pyramid spatial information into account* [63, 83]. Besides, we empirically observe that prompt-related methods perform not well when the number of patch tokens grows up from image to video scale, as shown in Figure 1.

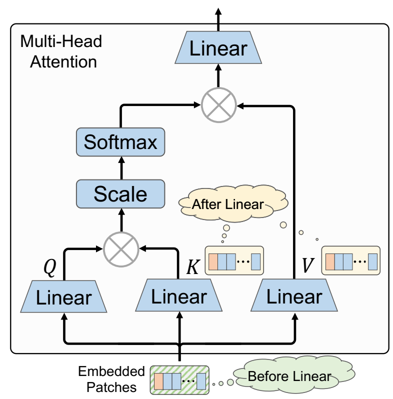


Figure 3: **Prompt tuning illustration.**

In summary, we present a strategy for tuning a pre-trained vision Transformer on a set of scalable vision recognition tasks (*e.g.*image domain and video domain). It adds limited learnable parameters for tuning while achieving comparable or even better performance than the full-tuning strategy. Moreover, AdaptFormer could serve as a generic module for a large variety of recognition tasks. 可以作为各种任务的通用模块

Insights of architecture design. The MLP module is important for ViTs. As illustrated in [30], MLPs prevent ViTs from producing a rank-1 matrix. Also, MLPs stop the ViT output from degenerations. Inspired by the above analysis, we believe an effective ViT adaptation shall focus on its MLPs rather than multi-head self attentions. Meanwhile, we learn from the inception framework [76] that parallel design is an effective way for feature ensemble. With the parallel design, the domain-specific features produced by the adapter module can supplement the domain-agnostic features from the fixed branch for a better feature ensemble. Our following experiments will verify that the parallel performs better than the sequential design.

Besides, though many advanced Transformer-based models [63, 83, 34, 90] which have emerged since the success of ViT having different attention mechanisms within the Transformer block, they all share the similar MLPs (feed-forward network) structures. Therefore, our AdaptMLP can be easily plugged into these ViT variants. Moreover, AdaptMLP can also be applied to more recent attention-free models [77, 61, 19].

4 Experiments

We evaluate the effectiveness of AdaptFormer by conducting extensive visual recognition experiments in both the image and video domains. We first describe our experimental settings in Sec. 4.1, covering the pre-trained backbones, baseline methods, downstream tasks and training details. We then compare AdaptFormer with baseline methods and provide a thorough analysis in Sec. 4.2. In addition, we also conduct ablation studies to explore different experimental configurations and explain what makes for the superiority of AdaptFormer in Sec 4.5.

4.1 Experimental Settings

Pre-trained backbone. We adopt the plain Vision Transformer (ViT) [31], *i.e.*, ViT-Base (ViT-B/16) as our backbone model and pre-train the model with both supervised and self-supervised approaches. Specifically, for **image**, we directly use the ImageNet-21k [26] supervised pre-trained model[3] and MAE [43] self-supervised model[4]. For **video**, we take both supervised and self-supervised pre-trained models from VideoMAE [78]. More details about pre-training approaches and datasets can be found in Appendix.

Initialization of AdaptFormer. For the original networks, we directly load the weights pre-trained on the upstream tasks and keep them frozen/untouched during the fine-tuning process. For the newly added modules, the weights of down-projection layers are initialized with Kaiming Normal [44], while the biases of the additional networks and the weights of the up-projection layers are configured with zero initialization. The reason for the zero initialization of other layers is that in this way, the initial newly added parameters are initialized such that the new function resembles the original one at the start of the fine-tuning stage. We empirically found that if the initialization deviates too far from the identity function, the model is not stable to train.

Baseline methods. We compare AdaptFormer with three commonly used fine-tuning approaches, including (1)*Linear probing:* adding an extra linear layer on top of the backbone and tuning the added parameters for evaluation. (2) *Full Fine-tuning:* setting all the parameters learnable and tuning them together. (3) *Visual Prompt Tuning (VPT):* [51] fine-tuning the extra token parameters as shown in Figure 3.

Downstream tasks. We evaluate our AdaptFormer on both image and video recognition tasks to verify its effectiveness. The specific datasets leveraged in this work are presented in the following.

- **Image domain :** CIFAR-100 [54] contains 50,000 training images and 10,000 validation images of resolution 32×32 with 100 labels. Street View House Numbers (SVHN) [37] is a digit classification benchmark dataset. In total, the dataset comprises over 600,000 labeled images, containing 73,257 training samples, 26,032 testing samples and 531,131 extra training data. The Food-101 [9] dataset consists of 101 food categories with a total of 101k images, including 750 training and 250 testing samples per category.

- **Video domain :** Something-Something V2 (SSv2) [39] is a large collection of video clips showing the people perform several normal actions in the daily life (*e.g.*, moving stuff and opening the door). It consists of 168,913 training samples, 24,777 validation samples and 27,157 testing samples, making a total of 220,847 videos with 174 labels. HMDB51 [55] is composed of 6,849 videos with 51 categories, making a split of 3.5k/1.5k train/val videos.

Implementation details. In this work, we use PyTorch toolkit [68] to conduct all experiments on NVIDIA V100 GPUs. Unless otherwise stated, we use 8×8 GPUs for video experiments and 1×8 GPUs for image experiments. Our default configurations follow the *linear probing* settings in [21, 43], which do *not* utilize many common regularization strategies, such as mixup [94], cutmix [91], color jittering and so on. More details can be found in Appendix.

4.2 Main Properties and Analysis

We compare the performance of different fine-tuning approaches in Table 1 with the backbones pre-trained via the self-supervised paradigms. The results show that AdaptFormer consistently

[3] https://github.com/rwightman/pytorch-image-models/releases/download/v0.1-vitjx/jx_vit_base_patch16_224_in21k-e5005f0a.pth

[4] https://dl.fbaipublicfiles.com/mae/pretrain/mae_pretrain_vit_base.pth

Table 1: **Fine-tuning with self-supervised pre-trained model.** For tunable parameters, we also report the parameter percentage in the brackets. Besides, we report the top-1 accuracy on different dataset with the absolute value and the gap value relative to the *full-tuning* regime. † denotes $0.1\times$ learning rate due to unstable training.

Method	Avg. Params (M)	Image			Video	
		CIFAR-100	SVHN	Food-101	SSv2	HMDB51
Full-tuning	86.04 (100%)	85.90	97.67†	90.09†	53.97	46.41
Linear	0.07 (0.08%)	69.83 (-16.07)	66.91 (-30.76)	69.74 (-20.35)	29.23 (-24.74)	49.84 (+3.43)
VPT [51]	0.08 (0.09%)	82.44 (-3.46)	94.02 (-3.65)	82.98 (-7.11)	43.73 (-10.24)	52.67 (+6.26)
AdaptFormer-1	0.10 (0.12%)	83.52 (-2.38)	93.04 (-4.63)	83.64 (-6.45)	50.03 (-3.94)	51.68 (+5.27)
AdaptFormer-4	0.15 (0.17%)	84.83 (-1.07)	96.19 (-1.48)	85.42 (-4.67)	54.70 (+0.73)	51.81 (+5.40)
AdaptFormer-64	1.26 (1.46%)	85.90 (0.00)	96.89 (-0.78)	87.61 (-2.48)	59.02 (+5.05)	55.69 (+9.28)

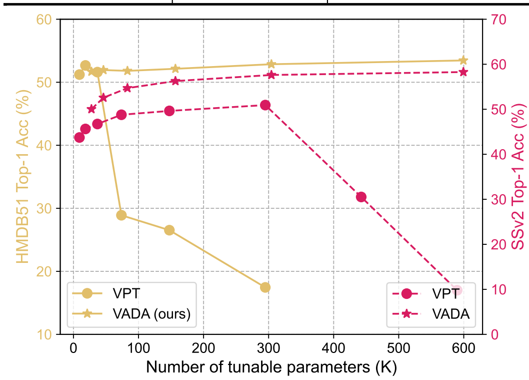


Figure 4: **The trend of performance as the number of tunable parameters grows up.** The accuracy of VPT drops dramatically when the parameter number exceeds task-specific value, while AdaptFormer is robust to the increasing parameters.

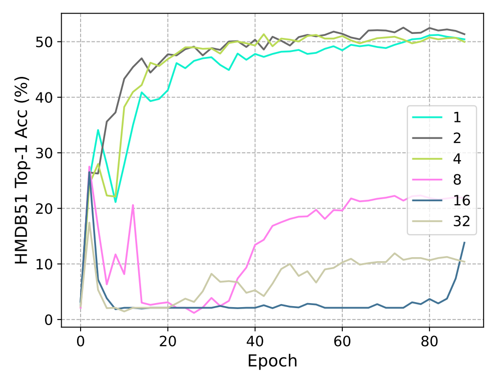


Figure 5: **Test accuracy of VPT [51] with different number of introduced tokens.** The optimization procedure becomes unstable when the token number is equal or larger than eight on HMDB51 dataset [55].

surpasses linear probing and Visual Prompt tuning (VPT) methods. Specifically, AdaptFormer-64 outperforms VPT on image benchmark CIFAR-100, SVHN, and Food-101, by 3.46%, 2.87%, and 4.63% respectively. On the more challenging video action recognition dataset Something-Something V2, the superiority becomes even more significant, *i.e.*, about 15%. Note that even compared with the full fine-tuning strategy, our AdaptFormer still outperforms by about 5% Top-1 accuracy on SSv2 dataset. To summarize, our AdaptFormer is highly parameter-efficient, as well as yielding good performance with parameter size at most 2% times than the full fine-tuning manner.

4.3 Scaling Tunable Parameters Up

Even though there are only limited parameters introduced, one might also argue that more tunable parameters of AdaptFormer contribute to its higher accuracy compared with VPT [51]. We conduct experiments to make a comprehensive discussion on this aspect.

As described in Sec. 3.3, the number of tunable parameters can be adjusted by changing the number of introduced tokens for VPT, or the hidden feature dimension for AdaptFormer. As shown in Figure 4, we conduct experiments with a wide range of tunable parameters on both SSv2 and HMDB-51 datasets. Since AdaptFormer and VPT share the same number of parameters of classification head on a specific dataset, we only report the tunable parameters on the x-axis, which comes from the visual prompts (VPT) or weight/bias of the down-up fully-connected layers (AdaptFormer), without calculating the parameters of classification head. For VPT, the number of introduced tokens is chosen from {1, 2, 4, 8, 16, 32, 48, 64}. Similarly, the number of hidden dimensions in AdaptFormer is in {1, 2, 4, 8, 16, 32}. AdaptFormer has a slight performance gain or maintains the accuracy stably when the parameters scale up. On the contrary, the performance of VPT decreases dramatically when the parameters exceed the task-specific value. Moreover, choosing the most suitable number of token

Table 2: **AdaptFormer for multi-label classification.**

Method	Params (M)	NUS-WIDE [24]
Full-tuning	85.86 (100%)	61.26
Linear	0.06 (0.08%)	51.19 (-27.25)
VPT [51]	0.07 (0.09%)	57.08 (-7.56)
AdaptFormer-1	0.09 (0.12%)	57.51 (-4.08)
AdaptFormer-4	0.15 (0.17%)	58.14 (-2.13)
AdaptFormer-64	1.25 (1.46%)	59.07 (-0.06)

number becomes laborious since it might be task-specific (*i.e.*varying from one dataset to the other one). For example, the accuracy of VPT keeps going up when the number of tunable parameters increases up to 300K on SSv2, whereas it begins to drop when the number of tunable parameters exceeds 50K on HMDB-51.

We further study the optimization procedures of VPT by monitoring the test accuracy of the training stage. As shown in Figure 5, we gradually increase the number of tokens in VPT and plot the Top-1 accuracy of each epoch. The training stages are stable when the number of tokens is less than or equal to 4, *e.g.*, $\{1, 2, 4\}$. However, when the number becomes 8 or larger, *e.g.*, $\{8, 16, 32\}$, the training procedure collapses at about the tenth epoch and achieves poor performance at the end of the training stage. On the contrary, the optimization procedures of AdaptFormer are stable when the number of parameters varies across a large range, as shown in Table 3a. The top-1 accuracy fluctuates within 1.5% when the number of parameters increases from 0.44M (`dim=16`) to 4.87M (`dim=256`).

4.4 Multi-Label Classification

We further conduct experiments on dataset with larger scale and diversity. Specifically, we evaluate AdaptFormer on NUS-WIDE [24] for multi-label classification. NUS-WIDE contains 269,648 images collected from Flicker, which are annotated with 81 visual concepts. Since some images are not available on Flicker, we only use 220,000 images following [7, 32]. We utilize mean average precision (mAP) as performance metric.

Settings and results. Our training settings mainly follow ASL [7]. Specifically, We trained all models for 40 epochs using Adam optimize and 1-cycle learning rate policy [73]. The maximal learning rate is 0.001. As shown in Table 2, though AdaptFormer-64 achieves a slightly lower mAP than fine-tuning, it significantly reduces the amount parameters that need to be updated (from 85.86 to 1.25M). Moreover, AdaptFormer has an clear advantage over other fine-tuning approaches including linear probing and VPT.

4.5 Ablation Studies

We ablate our AdaptFormer to study what properties make for a good AdaptFormer and observe several intriguing properties. The ablation studies conducted in this work are all performed on the SSv2 validation set [39].

Table 3: **AdaptFormer ablation experiments** with ViT-B/16 on SSv2. We report the top-1 accuracy on the `val` set. Most suitable settings are marked in color .

(a) **Middle dimension $\hat{d}$.**

mid dim	#params	top-1
1	0.16M	50.03
16	0.44M	57.62
32	0.73M	58.27
64	1.32M	**59.02**
256	4.87M	58.87

(b) **AdaptMLP inserted layers and form.**

layers	form	#params	top-1
$1 \rightarrow 6$	parallel	0.73	50.48
$7 \rightarrow 12$	parallel	0.73	57.99
$1 \rightarrow 12$	parallel	1.32	59.02
$1 \rightarrow 12$	sequential	1.32	58.17

(c) **Scaling factor s.**

factor	top-1
0.01	53.44
0.05	58.85
0.10	59.02
0.20	58.89

Middle dimension. The middle dimension controls the number of introduced parameters by AdaptFormer. Lower middle dimensions introduce fewer parameters with a possible performance cost. We ablate AdaptFormer on the middle feature dimension to study this effects. As shown in Table 3a, the accuracy consistently improves when the middle dimension increases up to 64 and reaches the saturation point when the middle dimension is about 64 on SSv2 dataset. We note that our AdaptFormer can achieve a decent performance when the middle dimension reduces even to one, about 50.03% top-1 accuracy.

We conduct more extensive ablation studies on middle dimension in Appendix Table 10 and found that the optimal middle dimension varies per dataset. For example, the accuracy reaches saturation when the middle dimension equals 64 on SSv2, whereas for NUS-WIDE dataset, the mAP slightly improves when the middle dimension increases from 64 to 512. However, AdaptFormer with middle dimension as 512 has 0.75 mAP higher (59.82 vs. 59.07 mAP) than the one with 64 at the cost of about 8 times more parameters. Therefore, we choose the `middle dimension=64` for both SSv2 and NUS-WIDE for a better trade-off.

Scaling factor. The scaling factor s is introduced to balance the *task-agnostic* features (generated by the original frozen branch) and the *task-specific* features (generated by the tunable bottleneck branch). We evaluate AdaptFormer with multiple s values and the results are summarized in Table 3c. Different from the scaling factor in NLP field which prefer s larger than 1 (*e.g.*, $s = 4$ in [42]), we empirically found that the s should be < 1 for vision tasks, otherwise the fine-tuning would become unstable. Besides, we found that AdaptFormer achieves optimal performance with $s = 0.1$. A larger or smaller s would bring slight performance drop. Thus, we choose $s = 0.10$ as a default setting.

AdaptFormer position. As shown in Table 3b, we further ablate on the specific position to introduce the AdaptMLP block. We gradually increase the number of AdaptMLP layers with a step of three (`start` $\rightarrow$ `end`, both included). We observe that the performance of AdaptFormer has a positive correlation with the number of added layers. In addition, AdaptFormer prefers the top part (the one far away from the input image) of the network to the bottom part when introducing the same number of layers, *e.g.*, AdaptFormer with `7` $\rightarrow$ `12` obtains over 14.5% higher accuracy than `1` $\rightarrow$ `6`, though both equipped with six AdaptMLP layers.

Insertion form. We study the insertion formulation by comparing the *parallel* and *sequential* instances which are illustrated in Figure 6. As shown in Table 3b, the parallel AdaptFormer is able to outperform the sequential one by 0.85% top-1 accuracy. The reason might be: **(1)** the parallel design maintains the original feature using an independent branch and aggregating updated context by element-wise scaled sum; **(2)** the sequential design is equivalent to adding more layers, which might cause optimization difficulty. Therefore, we adopt the parallel design as our default setting due to its superiority.

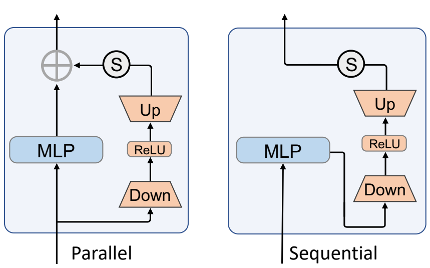


Figure 6: **Illustration of the *parallel* and *sequential* insertion form**. Comparison results are shown in Table 3b.

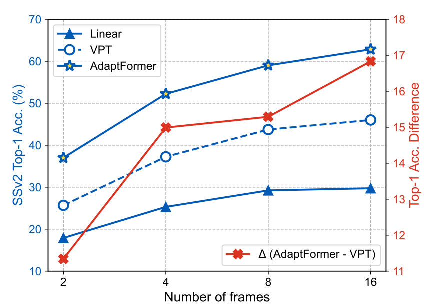


Figure 7: **Performance with video frames number.** AdaptFormer outperforms VPT and linear fine-tuning.

Number of frames. The number of embedded patch tokens increases linearly with the number of video frames for the plain ViT [31]. We conduct experiments with the different number of frames, *i.e.*, {2, 4, 8} and the results are shown in Figure 7. We observe that increasing the number of frames is beneficial for all these three fine-tuning methods. However, AdaptFormer consistently outperforms the linear manner (*e.g.*, +30% top-1 accuracy on 8 input frames) and VPT method(*e.g.*, +14% top-1 accuracy on 8 input frames).

4.6 Towards Visual Recognition Generalist Agent

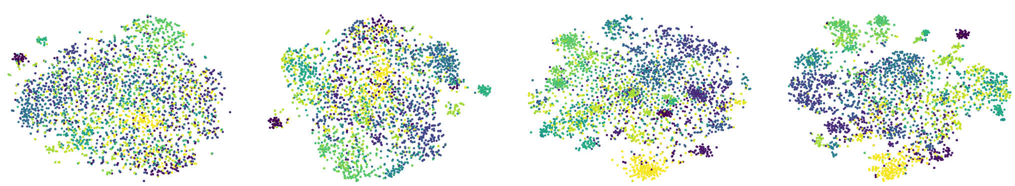
In the above experiments, we typically utilize a modality-specific pre-trained checkpoint for the corresponding downstream tasks. For example, we use Kinetics-400 (**video domain**) pre-trained model for downstream video action recognition on Something-Something V2 and HMDB-51 benchmarks. Besides, we use ImageNet-21K (**image domain**) pre-rained model for downstream image classification on CIFAR-100, SVHN and Food-101 benchmarks. Our AdaptFormer achieves superior performances in this *same network with modality-specific weights* scenario. Next, we take a further step to ask what would happen if using *the same network with the modality-agnostic weights* for multiple tasks in the multi-modalities downstream tasks?

Table 4: **Fine-tuning on video data with image** pre-trained model.

Method	Avg. Params (M)	Fine-tuning SSv2
Full-tuning	86.36	41.50
Linear	0.15	6.56
VPT [51]	0.16	16.94
AdaptFormer	1.33	46.06

We use the model pre-trained on ImagNet-21k to do action recognition on SSv2. As shown in Table 4, AdaptFormer is robust to domain shift caused by modality. The experimental results show that the linear probe approach obtains a very poor accuracy (*i.e.*, 6.56% top-1 accuracy) when fine-tuning on SSv2. Meanwhile, VPT [51] achieves a better performance than linear probe but it is not decent (*i.e.*, 16.94% top-1 accuracy). Our AdaptFormer, compared to the above two methods, attains a promising 46.06% top-1 accuracy, which is even higher than the full-tuning schedule (+4.56%).

4.7 Visualization



(a) Linear (0.08%) (Top1 29.23%) (b) VPT (0.09%) (Top1 43.73%) (c) Full fine-tune (100%) (Top1 53.97%) (d) AdaptFormer (1.26%) (Top1 59.02%)

Figure 8: **t-SNE visualizations on SSv2 val dataset.** We extract the final classification features from the top linear layer for t-SNE visualizations. The top-1 accuracy is reported in red, while the relative parameter (compared to the full fine-tuning strategy) is reported in blue.

To evaluate the quality of the produced features, we conduct t-SNE [80] visualizations on AdaptFormer and other baseline methods. The features are extracted from the SSv2 validation set via the ViT-Base backbone. Figure 8 shows that the linear fine-tuning and the VPT methods tend to output mixed features as shown in Figure 8(a)-(b). Compared with the above two methods, the full fine-tuning strategy performs well in projecting features. However, it consumes huge computational sources to tune the whole network parameters. Figure 8(d) validates that our AdaptFormer facilitates ViT-Base in generating more separable representations with fewer learnable parameters.

5 Conclusion

We present a conceptually simple yet effective framework, AdaptFormer, for efficiently adapting a pre-trained Vision Transformer (ViT) backbone to scalable vision recognition tasks. By introducing AdaptMLP, our AdaptFormer is able to fine-tune the lightweight modules for producing features adapted to multiple downstream tasks. The extensive experiments on five datasets, covering both the image and the video domains, validate that our proposed methods are able to increase the ViT's transferability with little computational cost. We hope our work will inspire future research in exploring more efficient fine-tuning methods for large vision models. One limitation is that AdaptFormer is only employed in recognition tasks in this work, it's unclear whether it can work well in tasks beyond recognition, *e.g.*, object detection and semantic segmentation. We leave it for the future exploration. Since our method is specially designed for efficient fine-tuning, we do not foresee obvious undesirable ethical/social impacts at this moment.

Acknowledgment. This work is supported by CCF-Tencent Open Fund. Ping Luo is supported by the General Research Fund of HK No.27208720, No.17212120, and No.17200622.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018. 4

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6836–6846, 2021. 3, 17

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

[4] 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. 2

[5] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 2

[6] Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. Simple, scalable adaptation for neural machine translation. *arXiv preprint arXiv:1909.08478*, 2019. 2

[7] Emanuel Ben-Baruch, Tal Ridnik, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification. *arXiv preprint arXiv:2009.14119*, 2020. 8

[8] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding. *arXiv preprint arXiv:2102.05095*, 2021. 3

[9] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014. 6

[10] 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*, 2020. 1, 3

[11] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, 2020. 1, 3

[12] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 2

[13] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *IEEE/CVF International Conference on Computer Vision*, 2021. 3

[14] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018. 18

[15] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 17

[16] Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022. 1

[17] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 3

[18] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021. 1

[19] Shoufa Chen, Enze Xie, Chongjian GE, Runjian Chen, Ding Liang, and Ping Luo. CycleMLP: A MLP-like architecture for dense prediction. In *International Conference on Learning Representations*, 2022. 5

[20] Shuo Chen, Tan Yu, and Ping Li. Mvt: Multi-view vision transformer for 3d object recognition. *arXiv preprint arXiv:2110.13083*, 2021. 3

[21] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *IEEE/CVF International Conference on Computer Vision*, 2021. 3, 6

[22] Cheng Chi, Fangyun Wei, and Han Hu. Relationnet++: Bridging visual representations for object detection via transformer decoder. *Advances in Neural Information Processing Systems*, 2020. 3

[23] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *NeurIPS 2021*, 2021. 5

[24] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, pages 1–9, 2009. 8

[25] Sudeep Dasari and Abhinav Gupta. Transformers for one-shot visual imitation. *arXiv preprint arXiv:2011.05970*, 2020. 1

[26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009. 3, 6, 17, 18

[27] 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. 1, 3

[28] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015. 17

[29] 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, 2021. 5

[30] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pages 2793–2803. PMLR, 2021. 5

[31] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 3, 4, 5, 6, 9, 17, 18

[32] Thibaut Durand, Nazanin Mehrasa, and Greg Mori. Learning a deep convnet for multi-label classification with partial labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 647–657, 2019. 8

[33] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *IEEE/CVF International Conference on Computer Vision*, 2021. 3

[34] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6824–6835, 2021. 5

[35] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. *arXiv preprint arXiv:2205.09113*, 2022. 2

[36] Chongjian Ge, Youwei Liang, Yibing Song, Jianbo Jiao, Jue Wang, and Ping Luo. Revitalizing cnn attention via transformers in self-supervised visual representation learning. *Advances in Neural Information Processing Systems*, 2021. 3

[37] Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013. 6

[38] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 17

[39] 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 *IEEE/CVF International Conference on Computer Vision*, 2017. 6, 8

[40] Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*, 2020. 3

[41] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 2021. 3

[42] 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*, 2022. 2, 9

[43] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021. 2, 3, 6, 17

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 6

[45] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 4

[46] 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*, 2019. 2, 3

[47] 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. *arXiv preprint arXiv:2106.09685*, 2021. 2, 3

[48] Drew A Hudson and Larry Zitnick. Generative adversarial transformers. In *International Conference on Machine Learning*, pages 4487–4499. PMLR, 2021. 1

[49] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 17

[50] Rishabh Jangir, Nicklas Hansen, Sambaran Ghosal, Mohit Jain, and Xiaolong Wang. Look closer: Bridging egocentric and third-person views with transformers for robotic manipulation. *IEEE Robotics and Automation Letters*, 2022. 1

[51] 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. 2, 3, 5, 6, 7, 8, 10, 18, 19

[52] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two pure transformers can make one strong gan, and that can scale up. *Advances in Neural Information Processing Systems*, 34, 2021. 1

[53] 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. 3

[54] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 6

[55] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *IEEE/CVF International Conference on Computer Vision*, 2011. 6, 7

[56] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021. 2, 3

[57] 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*, 2022. 3, 21

[58] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021. 2, 3, 5

[59] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *IEEE/CVF International Conference on Computer Vision*, 2021. 3

[60] Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. *arXiv preprint arXiv:2202.07800*, 2022. 3

[61] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34:9204–9215, 2021. 5

[62] Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021. 2

[63] 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*, 2021. 1, 3, 5, 18

[64] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021. 17, 18

[65] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 17

[66] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *European Conference on Computer Vision*, 2018. 3

[67] Tian Pan, Yibing Song, Tianyu Yang, Wenhao Jiang, and Wei Liu. Videomoco: Contrastive video representation learning with temporally adversarial examples. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 3

[68] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 6, 21

[69] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020. 2

[70] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*, 2020. 2

[71] 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*, 2021. 3

[72] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021. 3, 19

[73] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018. 8

[74] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *IEEE/CVF International Conference on Computer Vision*, 2017. 3

[75] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013. 17

[76] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 5

[77] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34:24261–24272, 2021. 5

[78] 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. 2, 3, 6, 17

[79] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 1

[80] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 10

[81] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 1, 3

[82] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019. 1

[83] 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*, pages 568–578, 2021. 1, 5

[84] Zhendong Wang, Xiaodong Cun, Jianmin Bao, and Jianzhuang Liu. Uformer: A general u-shaped transformer for image restoration. *arXiv preprint arXiv:2106.03106*, 2021. 3

[85] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. *arXiv preprint arXiv:2112.09133*, 2021. 2

[86] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 2021. 1, 3

[87] Ruihan Yang, Minghao Zhang, Nicklas Hansen, Huazhe Xu, and Xiaolong Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. In *International Conference on Learning Representations*, 2022. 1

[88] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019. 1

[89] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 17

[90] 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*, pages 558–567, 2021. 1, 5

[91] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *IEEE/CVF International Conference on Computer Vision*, 2019. 6

[92] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021. 2, 3

[93] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *CVPR*, 2022. 1

[94] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 6

[95] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. 3

[96] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021. 1

[97] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *International Conference on Learning Representations (ICLR)*, 2022. 2

[98] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 1, 3

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Shown in **Conclusion** Section.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Shown in **Conclusion** Section.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] As a URL shown in the abstract.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Shown in supplementary materials.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see Section 4.1
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] Shown in supplementary materials.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We used publicly available datasets whose licenses allow research usage.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] To the best of our knowledge, the data we used contains no personally identifiable information or offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

In this supplementary material, we will include the details about the pre-training and fine-tuning processes, the extensive experiments of AdaptFormer on hierarchical vision transformers (*e.g.*, AdaptFormer-Swin), and the pseudo-code of AdaptMLP in a PyTorch-like style.

A.1 Experimental Settings

A.1.1 Pre-training Approaches

Image. We use MAE [43] as our self-supervised pre-training method in the image domain, a simple yet effective method that first masks nearly 75% patches of the input image and then reconstructs the missing pixels. Specifically, we directly adopt the checkpoint[5] of ViT-B/16 for convenience, which is pre-trained on ImageNet-1K [26] for 800 epochs.

Video. We use VideoMAE [78] as our self-supervised pre-training method in the video domain, which is an direct extension of MAE to the video domain. VideoMAE utilizes the plain ViT [31] architecture of joint space-time attention mechanism [2, 64] and an extremely high proportion of masking ratio (*i.e.*, 90% to 95%) for pre-training. We also directly use the publicly available checkpoint[6], which is pre-trained on Kinetics-400 [15].

A.1.2 Implementation Details of Fine-tuning

Table 5: **Fine-tuning settings.** We present the shared configurations, like the optimizer and the base learning rate, the upper part, and show the seperated ones in the lower part.

Configuration	Image	Video
optimizer	SGD	
base learning rate	0.1	
weight decay	0	
optimizer momentum	0.9	
batch size	1024 images/frames	
learning rate schedule	cosine decay [65]	
GPU numbers	8	64
warmup epochs	20	10
training epochs	100	90
augmentation	RandomResizedCrop [43]	MultiScaleCrop [78]

The implementation details are summarized in Table 5. The video experiments are conducted on 64 Tesla V100 GPUs, while the image experiments are performed on 8 Tesla V100 GPUs. For the optimizer, different from [89] that adopts LARS, we leverage SGD [75] for stable training on small-scale dataset (*e.g.*, CIFAR10). The actual learning rate is calculated by: $lr = base_lr \times$batchsize / 256 following the linear lr scaling rule [38]. More detailed training configurations are presented in Table 5, including the batchsize, learning rate schedule and etc.

The experimental settings of image and video mainly follow the ones utilized in MAE [43] and VideoMAE [78], respectively. We insert an extra BatchNorm layer [49] without affine transformation (*i.e.* `affine=False`) before the final fully connected layer, following the common practice to normalize the pre-trained features [28, 43]. In addition, there is *no* flip augmentation during the fine-tuning stage for video data.

[5] `https://dl.fbaipublicfiles.com/mae/pretrain/mae_pretrain_vit_base.pth`

[6] `https://drive.google.com/file/d/1tEhLyskjb755TJ65ptsrafUG2llSwQE1/view?usp=sharing`

A.2 More Supplementary Results

A.2.1 AdaptFormer with Supervised Pre-training

In addition to the self-supervised pre-training presented in the main paper, we also evaluate AdaptFormer with the supervised pre-trained model. The results in Table 6 show that AdaptFormer still outperforms linear probe and VPT obviously. In addition, AdaptFormer surpasses full-tuning on four benchmarks (CIFAR100, SVHN, SSv2, HMDB51) with only 1.46% parameters. On the remaining benchmark (Food-101), AdaptFormer achieves an almost comparable performance to full-tuning (90.89% *v.s.* 90.96%).

Table 6: **Fine-tuning with *supervised* pre-trained model.** We report the tunable parameters percentage in the brackets. Besides, we report the top-1 accuracy on different dataset with the absolute value and the gap value relative to the *full-tuning* regime.

Method	Avg. Params (M)	Image			Video	
		CIFAR-100	SVHN	Food-101	SSv2	HMDB51
Full-tuning	86.04 (100%)	89.12	95.41	90.96	53.62	59.38
Linear	0.07 (0.08%)	85.95 (-3.17)	55.36 (-40.05)	88.14 (-2.82)	35.49 (-18.13)	70.31 (+10.93)
VPT [51]	0.08 (0.09%)	90.97 (+1.85)	92.77 (-2.64)	90.16 (-0.80)	55.22 (+1.60)	71.56 (+12.18)
AdaptFormer-64	1.26 (1.46%)	91.86 (+2.73)	97.29 (+1.88)	90.89 (-0.07)	60.18 (+6.56)	73.21 (+13.83)

A.2.2 AdaptFormer on Swin Transformer

Settings. We further demonstrate the effectiveness of AdaptFormer on hierarchical vision transformers, *e.g.*, Swin [63, 64]. We name AdaptFormer applied to Swin as *AdaptFormer-Swin*, to distinguish plain AdaptFormer (without any suffix) which is applied to the vanilla ViT [31]. It is noted that we can adopt AdaptMLP to Swin easily without any special modification as Swin and ViT share the same MLP architecture. However, VPT [51] needs additional handcraft designs to be suitable for the shifted local windows in the prevalent hierarchical vision transformers, which hinders its general applications.

We utilize Swin-B [63] and the video counterpart [64] for image and video. Similarly, we also directly use the officially provided checkpoints[7], which are pre-trained on ImageNet-21K [26] and Kinetics-600 [14].

Table 7: **Fine-tuning with *Swin* Transformer**. We utilize Swin-B [63] and Video Swin-B [64] for **image** and **video** experiments, respectively. Parameter percentage and performance difference are reported relative to *full-tuning* schedule.

Method	Avg. Params (M)	Image			Video	
		CIFAR-100	SVHN	Food-101	SSv2	HMDB51
Full-tuning	87.19 (100%)	89.95	97.03	91.43	52.92	68.73
Linear	0.11 (0.13%)	89.07 (-0.88)	69.06 (-27.97)	90.64 (-0.79)	28.32 (-24.61)	74.00 (+5.27)
AdaptFormer-Swin	1.25 (1.43%)	91.88 (+1.93)	97.31 (+0.28)	91.86 (+0.43)	54.09 (+1.17)	74.65 (+5.92)

Results. Since VPT is not applicable in Swin, we do not report its performance. Table 7 shows AdaptFormer-Swin performs well compared with other tuning strategies. For image benchmarks, our method can outperform full-tuning approach with only 1.43% parameters. Moreover, AdaptFormer-Swin surpasses linear probing by a significant margin, especially on the challenging dataset, SSv2. The results validate that AdaptFormer is able to generally boost the transferability of various vision Transformer variants.

[7]Image: https://github.com/SwinTransformer/storage/releases/download/v1.0.4/swin_base_patch244_window877_kinetics600_22k.pth
Video: https://github.com/SwinTransformer/storage/releases/download/v1.0.0/swin_base_patch4_window7_224_22k.pth

A.3 Possible Architectures

We explore other possible architectures utilized in AdaptFormer. Specifically, we further replace the MLP architectures within the AdaptMLP module by the convolution layer, depthwise convolution layer, and LayerNorm layer. For fair comparisons, we carefully design the above modules to meet the comparable number of parameters ($\tilde{1}$.3M). The experimental results of different adapter modules are shown in Table 8, which validates that the simple MLP modules are simple yet effective compared with the other architectures. For example, our AdaptMLP module surpasses the AdaptConv module by 0.55% Top1 accuracy on SSv2 dataset.

Table 8: **Fine-tuning with different adapter modules**. We use AdaptConv to denote the designed adapter module with convolution layers, while AdaptDepthwise-Conv is utilized to denote the designed adapter module with depthwise convolution layers. Besides, we also replace the MLP architectures with LayerNorm layer as AdaptLayerNorm-In.

Methods	Avg Parameters	SSv2 Top1	NUS-WIDE mAP	CIFAR100 Top1
AdaptMLP	1.28	59.02	59.07	85.93
AdaptConv	1.39	58.47	58.86	85.42
AdaptDepthwise-Conv	1.29	58.15	58.73	85.37
AdaptLayerNorm-In	1.30	57.85	58.51	85.71

A.4 Evaluation on ImageNet-1k datasets

Table 9: **Fine-tuning with AdaotFormer on ImageNet-1k dataset**. We load the weights pretrained on ImageNet-21K and evaluate the classification performance on ImageNet-1K.

Methods	Parameters (M)	ImageNet-1k Top1 (%)
Full Fine-tuning	86.57	82.26
Liner	0.77	80.95
VPT	0.78	81.68
AdaptFormer-1	0.80	82.33
AdaptFormer-4	0.85	82.26
AdaptFormer-16	1.07	82.24
AdaptFormer-64	1.96	81.86

We point out that in order to evaluate the adaptation performance across datasets, it's an unreasonable setting to fine-tune the ImageNet-1k dataset with the ImageNet-21k pre-trained weights. This is because **ImageNet-1K** is a subset of the **ImageNet-21K** as introduced in [72]. In contrast, in all the previous experiments, there is no overlap between the fine-tuning and pre-trained datasets. However, we document the experiments of fine-tuning with the ImageNet-1k dataset for the completeness.

We adopt exactly identical training configurations to conduct experiments in this subsection. We experiment with `middle dimension = {1, 4, 16, 64}` on ImageNet-1K, and the results are shown Table 9.

Results. Comparing the results of AdaptFormer with different `middle dimension` ({1, 4, 16, 64}) on ImageNet-1K, we find that AdaptFormer with the smallest number of parameters (**AdaptFormer-1**) achieves the best top-1 accuracy (82.33%). Furthermore, when the 'middle dimension' increases from 1 to 4 or 16, AdaptFormer has a slight performance drop (AdaptFormer-4 (-0.07%) and AdaptFormer-16 (-0.09%)). Further increasing the `middle dimension` to 64 will cause a relatively clear performance drop (-0.47%).

Discussions. Although our AdaptFormer-64 does not have a clear advantage compared with VPT [51], our AdaptFormer-1 outperforms VPT by +0.65% top-1 accuracy with only 0.02M additional parameters. Besides, the trend of *classification accuracy* changing with `middle dimension` on ImageNet-1k is different from other datasets in our paper, *e.g.*, AdaptFormer with `middle dimension=64` achieves better top-1 accuracy than with `middle dimension=1` on CIFAR-100. We empirically find introducing a small number of parameters (AdaptFormer-1) is sufficient for ImageNet-1K fine-tuning, while more introduced parameters will make a larger change to the original model and make it harder for optimization since ImageNet-1K is a subset of ImageNet-21K. However, for other datasets (e.g., CIFAR-100) with no overlap between the fine-tuning datasets and the pre-trained datasets, more introduced parameters are needed for learning better domain knowledge.

A.5 Extended experiments on middle dimension

We conduct the extended ablation studies on the middle dimension design in this sub-section. We aim to seek for a trade-off between model capacity (i.e., potential) and adaptation efficiency. In fact, the middle dimension has a main influence on the parameter size of adapter. The higher dimension brings more parameters while the efficiency and storage are limited. As shown in Table 10, we evaluate several numbers of middle dimension and found that using 64 is optimal to achieve accuracy, light-weight storage, and efficiency.

Table 10: **AdaptFormer ablation experiments** with ViT-B/16 on SSv2. The experimental results on middle dimension are investigated.

Middle Dimension	Parameters (M)	SSv2 Top1 (%)	NUS-WIDE mAP (%)
1	0.16	50.03	57.51
4	0.22	54.70	58.14
16	0.44	57.62	59.00
32	0.73	58.27	59.09
64	1.32	59.02	59.07
128	2.51	58.95	59.49
256	4.87	58.87	59.62
512	9.59	58.98	59.82

A.6 Analysis on the fine-tuning time and inference latency

To analysis the computational efficiency, we compare the fine-tuning time and inference time on a single NVIDIA A100-40G GPU. We utilize SSv2 video classification for this part. For fine-tuing, we experiment with batchsize of 32. For inference, we test the latency with multiple batch sizes to get a comprehensive comparison under various inference scenarios. All the time is measured in milliseconds averaged over 100 trials. The results are summarized in Table 11 and Table 12. As shown in Table 11, AdaptFormer only costs less than a half of the fine-tuning time compared with the full-tuning. Moreover, AdaptFormer significantly outperforms linear probing in terms of accuracy with a slight longer fine-tuning time. For inference, AdaptFormer introduce a negligible FLOPs and latency compared with the Linear/Full-tuning.

Table 11: **Fine-tuning time of a single forward-backward step averaged over 100 trials.**

Methods	Latency (B=32)
Full-tuning	355.0 ms
Linear	140.2 ms
VPT	210.3 ms
AdaptFormer	162.2 ms

Table 12: **Inference time of a single forward step averaged over 100 trials.**

Methods	Flops (B=1)	Latency (B=1)	Latency (B=16)	Latency (B=32)
Linear/Full-tuning	78.915G	11.1 ms	22.4 ms	42.3 ms
VPT	79.029G	11.3 ms	22.9 ms	42.4 ms
AdaptFormer	79.840G	11.9 ms	23.2 ms	42.8 ms

A.7 Discussion about ImageNet and Kinetics Pre-training

The type of spatiotemporal attention (**divided** *vs.* **joint**) determines whether the performance of the model pre-trained on ImageNet can outperform the model pre-trained on Kinetics.

A similar phenomenon has been discussed in recent work, Uniformer [57], independently. We borrow the experimental results from Table 4(c) in Uniformer paper [57]. Specifically, the **divided** spatiotemporal attention prefers ImageNet to Kinetics-400 for the pre-training dataset. The performance of the **divided** attention model pre-trained on ImageNet outperforms the model pre-trained on Kinetics-400. On the contrary, the **joint** spatiotemporal attention prefers Kinetics-400 to ImgaeNet. The **joint** attention model attains higher top-1 accuracy with Kinetics-400 pretraining compared to ImageNet (53.8 *vs.* 52.0).

We adopt the **joint** spatiotemporal attention for all video-related experiments in this work (introduced in Appendix A.1.1). Therefore, our experimental phenomenon is consistent with the joint attention in [57], i.e., Kinetics pretraining is preferable.

A.8 Implementation

Algorithm 1 Implementation of AdaptMLP in PyTorch-like style.

```python
class AdaptMLP(nn.Module):
    def __init__(self, original_mlp, in_dim, mid_dim, dropout=0.0, s=0.1):
        super().__init__()
        self.original_mlp = original_mlp # original MLP block
        # down --> non linear --> up
        self.down_proj = nn.Linear(in_dim, mid_dim)
        self.act = nn.ReLU()
        self.up_proj = nn.Linear(mid_dim, in_dim)
        self.dropout = nn.Dropout(dropout)
        self.scale = s # scaling factor
        # initialization
        nn.init.kaiming_uniform_(self.down_proj.weight)
        nn.init.zeros_(self.up_proj.weight)
        nn.init.zeros_(self.down_proj.bias)
        nn.init.zeros_(self.up_proj.bias)
        # freeze original MLP
        for _, p in self.original_mlp.named_parameters():
            p.requires_grad = False

    def forward(self, x):
        down = self.down_proj(x)
        down = self.act(down)
        down = self.dropout(down)
        up = self.up_proj(down)
        output = self.original_mlp(x) + up * self.scale
        return output
```

The core part of AdaptFormer is replacing the original MLP with AdaptMLP, which consists of the frozen original MLP and newly introduced `Down` $\rightarrow$ `ReLU` $\rightarrow$ `Up` layers, which are tunable at the fine-tuning stage. Algorithms 1 provides the implementation of AdaptMLP written in PyTorch [68].

For more implementation details, please refer to the provided source code.