ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240489

Convolutional Bypasses Are Better Vision Transformer Adapters

Shibo Jie[a,*], **Zhi-Hong Deng**[a], **Shixuan Chen**[b] **and Zhijuan Jin**[b]

[a]State Key Laboratory of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University
[b]China Tower Corporation Limited

Abstract. The pretrain-then-finetune paradigm has been widely adopted in computer vision. But as the size of *Vision Transformer* (ViT) grows exponentially, the full finetuning becomes prohibitive in view of the heavier storage overhead. Motivated by *parameter-efficient transfer learning* (PETL) on language transformers, recent studies attempt to insert lightweight adaptation modules (e.g., adapter layers or prompt tokens) to pretrained ViT and only finetune these modules while the pretrained weights are frozen. However, these modules were originally proposed to finetune language models and did not take into account the prior knowledge specifically for visual tasks. In this paper, we propose to construct *Convolutional Bypasses* (Convpass) in ViT as adaptation modules, introducing only a small amount (less than 0.5% of model parameters) of trainable parameters to adapt the large ViT. Different from other PETL methods, Convpass benefits from the hard-coded inductive bias of convolutional layers and thus is more suitable for visual tasks, especially in the low-data regime. Experimental results on VTAB-1K benchmark and few-shot learning datasets show that Convpass outperforms current language-oriented adaptation modules, demonstrating the necessity to tailor vision-oriented adaptation modules for adapting vision models.

1 Introduction

Pretraining on large-scale datasets (e.g., ImageNet) and then fully finetuning on downstream tasks has become the de-facto paradigm to achieve state-of-the-art (SOTA) performance on visual tasks [21]. However, this paradigm is not storage-efficient – it requires one to store a whole model for each downstream task. Recently, as *Vision Transformer* (ViT) [8] dominates vision field gradually, the size of vision models has grown exponentially (58M of ResNet-152 [10] vs. 1843M of ViT-G [43]), which creates the demand for *parameter-efficient transfer learning* (PETL) on ViT.

Fortunately, since transformer was first adopted in *neural language processing* (NLP) [35], PETL on large pretrained language models has been studied sufficiently [14, 15, 23, 9], which can be easily ported to ViT. Concretely, these PETL methods insert lightweight adaptation modules into the pertrained models, freeze the pretrained weights, and finetune these modules end-to-end to adapt to downstream tasks. Recent work has verified the effectiveness of these PETL methods on ViT [16, 46], but we raise a question: *Are these modules designed for the language models optimal for vision models as well?*

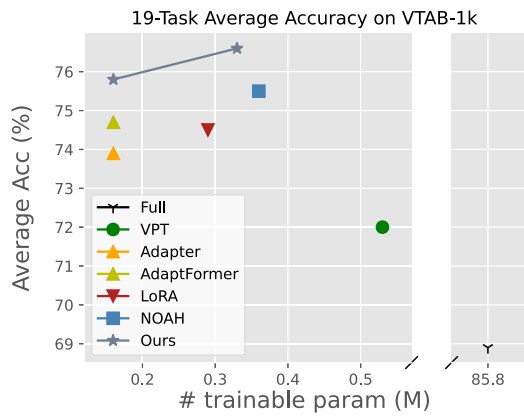


Figure 1: **Average accuracy vs. number of trainable parameters on VTAB-1K benchmark.** Our vision-oriented Convpass outperforms other language-oriented methods.

It is known that NLP tasks and visual tasks desire different inductive bias, which profoundly affects the model architecture design. By analyzing current PETL methods from an unraveled perspective, we argue that these methods, called "language-oriented modules", also imply the inductive bias for language, e.g., weak spatial relation and support for variable-length input. Therefore, a better adaptation module for ViT should also reflect visual inductive bias, such as spatial locality and 2D neighborhood structure, which is referred to as "vision-oriented modules".

When a model (e.g., ViT) has weak inductive bias, it needs a large amount of data to learn the inductive bias from scratch. This may not be a serious problem in the pretraining process, since we can leverage easily accessible unlabeled data for self-supervised learning [2, 11], or resort to multi-modal pretraining [32, 40]. However, data of downstream tasks is usually collected from specific domains that may be expensive or hard to acquire. Therefore, besides the inductive bias learned from pretraining data, a well-designed vision-oriented PETL module is expected to introduce additional inductive bias and improve data efficiency much further.

In this paper, we propose to construct *Convolutional Bypasses* (Convpass) in ViT as adaptation modules. Convpass is an inserted convolutional bottleneck block parallel to the MHSA or MLP block, which "bypasses" the original ViT block. It reconstructs the spatial

* Email: parsley@pku.edu.cn

structure of the token sequence and performs convolution on image tokens and `[cls]` token individually. During finetuning, only these Convpass modules and the classification head are updated. Due to the hard-coded locality of convolutional layers, Convpass can capture visual information more efficiently, especially when the downstream data is limited. As shown in Figure 1, Convpass only introduces and tunes about 0.33M new parameters for a ViT-B of 86M, while achieving better performance than both full finetuning and current SOTA language-oriented methods on 19-task VTAB benchmark [42]. Further experiments on few-shot learning demonstrate that Convpass also outperforms other baselines in the low-data regime, and can be directly used on vision-language model [32] with good domain generalization performance.

We summarize the contributions as follows:

- We point out the weak visual inductive bias of current PETL methods that limits their performance on ViT.
- We propose Convpass, a simple yet effective PETL method which leverages trainable convolutional blocks as bypasses to adapt pretrained ViT to downstream visual tasks.
- Experimental results show that Convpass outperforms previous language-oriented methods, indicating the necessity to tailor vision-oriented adaptation modules for vision models.

2 Related Work

2.1 Vision Transformer

Transformer-based models have achieved great success in NLP [7, 33, 4]. ViT adopts this architecture in visual tasks by partitioning the images into patches which are embedded and flattened into 1D token sequences.

In ViT, each layer consists of two kinds of blocks: *Multi-Head Self-Attention* (MHSA) and *Multi-Layer Perceptron* (MLP). In an MHSA block, the input sequence $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ is firstly projected to query $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_q$, key $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_k$, and value $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_v$, respectively, in which $\boldsymbol{W}_{q/k/v} \in \mathbb{R}^{d\times d}$. They are further divided into N_h heads: $\{\boldsymbol{Q}^{(i)}\}_{i=1}^{N_h}, \{\boldsymbol{K}^{(i)}\}_{i=1}^{N_h}, \{\boldsymbol{V}^{(i)}\}_{i=1}^{N_h}$. Then, the self-attention of a single head is formulated as

$$\mathit{Attn\text{-}Head}^{(i)}(\boldsymbol{X}) = \mathit{Softmax}\left(\frac{\boldsymbol{Q}^{(i)}\boldsymbol{K}^{(i)\intercal}}{\sqrt{d}}\right)\boldsymbol{V}^{(i)}$$

The outputs of all heads are further concatenated and linearly projected as the outputs of the MHSA block.

An MLP block consists of two fully-connected (FC) layers, whose weights are $\boldsymbol{W}_1 \in \mathbb{R}^{d\times D}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{D\times d}$, respectively. Ignoring the bias parameters for simplicity, the MLP is formulated as

$$\mathit{MLP}(\boldsymbol{X}) = \mathit{GELU}(\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{W}_2$$

Since ViT has much less visual inductive bias, it performs worse than its convolutional counterparts (e.g., ResNet) when the training data is not sufficient. For this reason, some recent work proposes to introduce visual inductive bias into ViT [26, 39], which significantly reduces its dependency on scale of dataset. However, vanilla ViT still has some nonnegligible advantages. Since vanilla ViT shares the same backbone as the transformer-based language models, it can leverage current SOTA multi-modal pretraining methods with a vast amount of auto-annotated image-text pairs [38, 40, 19]. Therefore, we still focus on PETL on vanilla ViT architecture, but propose to introduce hard-coded inductive bias by adaptation modules during finetuning instead of pretraining.

2.2 Parameter-Efficient Transfer Learning

PETL aims at using a small number of trainable parameters to adapt large models to downstream tasks. We here introduce some common PETL methods used for ViT.

Adapter [14, 31, 18, 20] is a bottleneck MLP block composed of two fully connected layers, whose weights are $\boldsymbol{W}_{down} \in \mathbb{R}^{d\times h}$ and $\boldsymbol{W}_{up} \in \mathbb{R}^{h\times d}$, where $h << d$. Adapters are inserted into networks as residual connections, i.e., given an input $\boldsymbol{X} \in \mathbb{R}^{N\times d}$, the computation is formulated as

$$\boldsymbol{X}' \leftarrow \boldsymbol{X} + \phi(\boldsymbol{X}\boldsymbol{W}_{down})\boldsymbol{W}_{up}$$

where ϕ is activation function such as GELU.

Pfeiffer et al. [31] propose to place Adapters after the MLP blocks (i.e., $\boldsymbol{X}$ is the output of MLP blocks), which has been proved to be an efficient design in previous literature [15], so we follow this setting in this paper. Besides the above design, He et al. [9] and Chen et al. [5] also propose a parallel Adapter to adapt MLP blocks, which is formulated as

$$\boldsymbol{X}' \leftarrow \boldsymbol{X} + \mathit{MLP}(\boldsymbol{X}) + s\cdot\phi(\boldsymbol{X}\boldsymbol{W}_{down})\boldsymbol{W}_{up}$$

where s is a hyperparameter, $\boldsymbol{X}$ is the input of MLP blocks. This Adapter design is referred to as **AdaptFormer** by Chen et al. [5].

LoRA [15] learns the low-rank approximation of increments of $\boldsymbol{W}_q$ and $\boldsymbol{W}_v$. Formally, it decomposes $\Delta\boldsymbol{W}_{q/v}$ into $\boldsymbol{A}_{q/v}\boldsymbol{B}_{q/v}$, where $\boldsymbol{A}_{q/v} \in \mathbb{R}^{d\times r}$, $\boldsymbol{B}_{q/v} \in \mathbb{R}^{r\times d}$ and $r << d$. The query and key are computed as

$$\boldsymbol{Q/V} = \boldsymbol{X}\boldsymbol{W}_{q/v} + s\cdot\boldsymbol{X}\boldsymbol{A}_{q/v}\boldsymbol{B}_{q/v}$$

in which s is a scaling hyperparameter.

VPT [16] has a similar idea with P-Tuning v2 [25]. It concatenates the input $\boldsymbol{X}$ with several trainable prompts $\boldsymbol{P} \in \mathbb{R}^{l\times d}$ before each layer. This extended sequence is formulated as

$$\boldsymbol{X}' \leftarrow [\boldsymbol{X}, \boldsymbol{P}]$$

These prompts are then cut away at the end of a layer, and the prompts for the next layer are concatenated.

NOAH [46] is a newly proposed PETL method for ViT, which combines the above three modules together and performs neural architecture search on hidden dimension h of Adapter, rank r of LoRA, and prompt length l of VPT.

Note that although VPT and NOAH are proposed for visual tasks, their components are ported from NLP in essence. Therefore, all the aforementioned PETL methods can be classified as language-oriented methods. Other PETL methods include BitFit [41], which finetunes the bias parameters only; Sidetune [44], which finetunes a small side-network and interpolates between pretrained and side-tuned features; FacT [17], which tensorizes the ViT as a single tensor and reparameterizes its change as several factors according to Tensor-Train or Tucker format; and SSF [24], which modifies the intermediate features of network via affine transformation.

3 Methodology

3.1 Rethinking Adapters from an Unraveled View

Since Adapters and MHSA/MLP blocks all contain skip connections, we can unravel the ViT and rewrite it as a collection of paths. Veit et al. [36] point out that the original network is an ensemble of unraveled paths, so we here give a look at these paths to analyze the property of the original network.

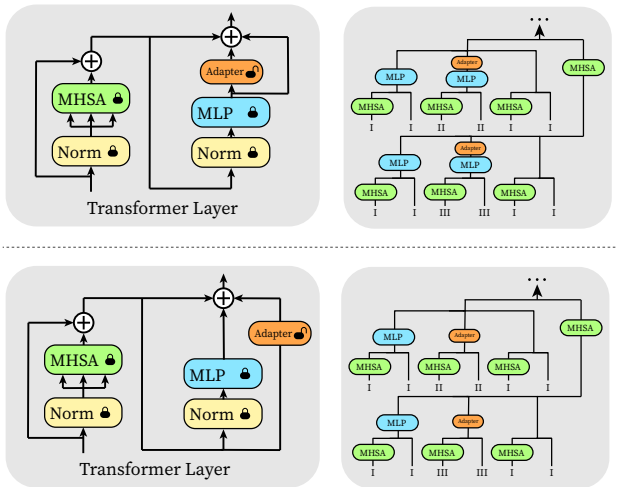


Figure 2: **Illustration of the unraveled view of ViT equipped with sequential Adapter (top) or parallel Adapter (bottom).** For simplicity, we show the unraveled view of a fragment of ViT (MHSA-MLP-MHSA) and the type of each path. Normalization layers are omitted.

As shown in Figure 2, a ViT equipped with Adapter can be viewed as an ensemble of three types of paths: (**Type I**) Frozen paths, which only contain MHSA/MLP blocks of the ViT. These paths are not trainable, and the sum of their outputs is identically equal to the output of the pretrained ViT. (**Type II**) MHSA-Adapter paths, where all MHSA blocks come before the first Adapter. (**Type III**) Adapter-MHSA paths, where at least one MHSA block is placed after an Adapter.

Finetuning the Adapters is equivalent to fitting the changes of outputs by the paths of **Type II & III**. In **Type II** paths, given the same input, the output tokens of the last MHSA blocks are unchanged, and there is no information exchange between tokens after that. Therefore, only the **Type III** paths, in fact, make changes to the token mixer of the pretrained ViT.

In a **Type III** paths, we can treat all Adapters and MLP blocks before an MHSA block as a part of its query/key/value transformation, i.e., complicate these transformations form linear mapping to

$$\boldsymbol{Q}/\boldsymbol{K}/\boldsymbol{V} = f_{q/k/v}(\boldsymbol{X})$$

where $f_{q/k/v}$ are channel-wise MLPs. Therefore, finetuning **Type III** paths can be considered as finetuning the MHSA with the complicated query/key/value transformations.

Meanwhile, since LoRA finetunes $\boldsymbol{W}_{q/v}$ in a low-rank subspace and VPT can be regarded as parallel and gated Adapters [9], all these language-oriented methods rely on tuning MHSA to adjust the token mixer on downstream tasks. MHSA, however, lacks visual inductive bias, which may perform poorly when the data of downstream visual tasks is limited.

3.2 *Adapting ViT via Convolutional Bypasses*

Recent studies on modifying the architecture of ViT have verified that introducing convolution into ViT will improve the performance when training data is not adequate [8, 39]. Since the data of downstream tasks is usually limited even few-shot, we can also introduce convolution into the adaptation modules for PETL.

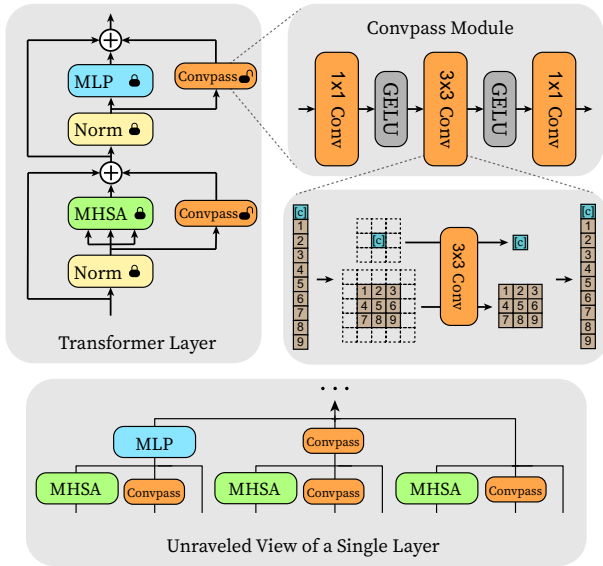


Figure 3: **Overview of the proposed method.** We restore the spatial structure of the token sequence, and use trainable ResNet-style convolutional blocks as bypasses. The `[cls]` token is regarded as an individual image.

As illustrated in Figure 3, a Convpass module consists of three convolutional layers: an 1×1 convolution reducing the channel, a 3×3 convolution with the same input and output channel, and an 1×1 convolution expanding the channel. Since ViT flattens the image into an 1D token sequence, we restore the 2D structure before convolution. The `[cls]` token serves as an individual image. The Convpass modules are placed parallel to the MHSA/MLP blocks, which can be formulated as

$$\boldsymbol{X}' \leftarrow \boldsymbol{X} + \mathit{MHSA/MLP}(\mathit{LN}(\boldsymbol{X})) + s \cdot \mathit{Convpass}(\mathit{LN}(\boldsymbol{X}))$$

where s is a hyperparameter and *LN* is Layer Normalization [1]. Note that the Convpass modules are similar to the *residual bottleneck blocks* of ResNet [10]. If we ignore the MHSA/MLP blocks, the ViT will turn into a ResNet-like CNN.

From the unraveled view, we can find that in each transformer layer, besides the frozen paths, there are also trainable paths that only contain Convpass or contain both Convpass and MHSA which act as token-mixers. Therefore, the original transformer layers are converted to an ensemble of transformers, ResNet-like CNNs, and hybrid models. This design can help transfer learning from several perspective. First, since all the trainable paths contain Convpass modules, the finetuning process can benefit from the inherent visual inductive bias of CNN. Second, the 2D neighborhood structure of the 3×3 convolution focuses on local information, complementary to the MHSA that has global receptive field.

Convpass is storage-efficient. If the bottleneck channel size (i.e., the input & output channel of the 3×3 convolution) is denoted as h, and the amount of ViT layers is L, the number of trainable parameters is $2L \times \big((2h+1)d + 9h^2 + 2h\big)$. In view of $h << d$ (e.g., $d = 768, h = 8$ in our experiments), this amount is $\mathcal{O}(Lhd)$, which is negligible compared to ViT's $\mathcal{O}(Ld^2)$ parameters.

3.3 *Reduce Redundancy of Convpass*

Inspired by the FacT [17], we infer that there is redundancy in the parameters of Convpass. Building on this insight, we further propose

		Natural							Specialized				Structured								
	# param (M)	Cifar100	Caltech101	DTD	Flower102	Pets	SVHN	Sun397	Camelyon	EuroSAT	Resisc45	Retinopathy	Clevr-Count	Clevr-Dist	DMLab	KITTI-Dist	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Ele	Average
Traditional Finetuning																					
Full	85.8	68.9	87.7	64.3	97.2	86.9	87.4	38.8	79.7	95.7	84.2	73.9	56.3	58.6	41.7	65.5	57.5	46.7	25.7	29.1	68.9
Linear	0	64.4	85.0	63.2	97.0	86.3	36.6	51.0	78.5	87.5	68.5	74.0	34.3	30.6	33.2	55.4	12.5	20.0	9.6	19.2	57.6
PETL methods																					
VPT-Shallow	0.063	77.7	86.9	62.6	97.5	87.3	74.5	51.2	78.2	92.0	75.6	72.9	50.5	58.6	40.5	67.1	68.7	36.1	20.2	34.1	67.8
VPT-Deep	0.53	**78.8**	90.8	65.8	98.0	88.3	78.1	49.6	81.8	**96.1**	83.4	68.4	68.5	60.0	46.5	72.8	73.6	47.9	32.9	37.8	72.0
Adapter	0.16	69.2	90.1	68.0	98.8	89.9	82.8	54.3	84.0	94.9	81.9	75.5	80.9	65.3	48.6	78.3	74.8	48.5	29.9	41.6	73.9
AdaptFormer	0.16	70.8	91.2	70.5	99.1	90.9	86.6	54.8	83.0	95.8	84.4	76.3	81.9	64.3	49.3	80.3	76.3	45.7	31.7	41.1	74.7
LoRA	0.29	67.1	91.4	69.4	98.8	90.4	85.3	54.0	**84.9**	95.3	84.4	73.6	**82.9**	**69.2**	49.8	78.5	75.7	47.1	31.0	44.0	74.5
NOAH	0.36	69.6	**92.7**	70.2	99.1	90.4	86.1	53.7	84.4	95.4	83.9	**75.8**	82.8	68.9	49.9	**81.7**	81.8	48.3	32.8	44.2	75.5
Convpass$_{share}$	0.056	69.8	89.3	71.7	99.1	**91.0**	88.1	53.7	85.0	95.8	84.0	75.4	81.5	63.9	50.2	80.6	84.5	51.8	32.6	40.7	75.4
Convpass$_{attn}$	0.16	71.8	90.7	72.0	99.1	**91.0**	89.9	54.2	85.2	95.6	83.4	74.8	79.9	67.0	50.3	79.9	84.3	**53.2**	34.8	43.0	75.8
Convpass	0.33	72.3	91.2	**72.2**	**99.2**	90.9	**91.3**	**54.9**	84.2	**96.1**	**85.3**	75.6	82.3	67.9	**51.3**	80.0	**85.9**	53.1	**36.4**	**44.4**	**76.6**

Table 1: **Full results on the VTAB-1K benchmark**. "Average" denotes the average results over three group-wise averages in Figure 4. "# params" denotes the average number of trainable parameters **in backbones**. Convpass(-attn) achieves 12 SOTA results out of the 19 tasks among PETL methods.

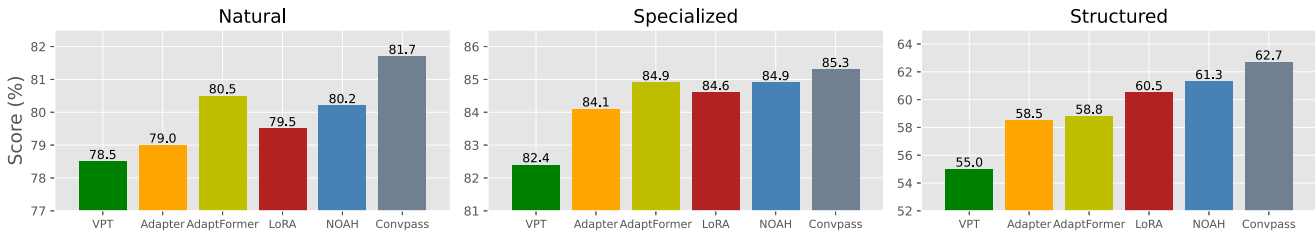


Figure 4: **Group-wise average results on VTAB-1K.** Convpass outperforms other baselines in all of the three groups.

a design to lightweight Convpass. Specifically, we share two 1×1 convolutions across all Convpass modules, and for the 3×3 convolutions, we employ depth-wise convolutions. The bias terms are deprecated in all convolutional layers. Then the number of trainable parameters is reduced to $2dh + 9Lh$. In view of $L << d$, the number of Convpass's trainable parameters is reduced from $\mathcal{O}(Lhd)$ to $\mathcal{O}(hd)$.

4 Experiments

4.1 Transfer Learning on VTAB-1K Benchmark

First of all, our method is evaluated on the basic transfer learning scenario – finetuning the pretrained models on various dowmstream tasks.

4.1.1 Datasets

To evaluate the performance on transfer learning of our methods, we use VTAB-1K [42] as a benchmark. VTAB-1K benchmark contains 19 image classification tasks from different fields, which can be roughly categorized into three groups: Natural, Specialized, and Structured. Each classification task only has 1,000 training samples, which are split into a training set (800) and a validation set (200) during hyperparameter search. The reported results are produced by evaluating the model trained on all the 1,000 training samples on test set. Following [46], we resize the images to 224×224, and then normalize them with ImageNet's mean and standard deviation.

4.1.2 Baselines

We compare our method with two traditional finetuning methods: **Full** finetuning, which optimizes all parameters end-to-end; **Linear** evaluation, which freezes the pretrained backbone and only learns a classification head; as well as four PETL methods: **VPT**, **Adapter**, **AdaptFormer**, **LoRA**, and **NOAH**. For our method **Convpass**, we also report two simplified variant: **Convpass$_{attn}$**, which only inserts the Convpass modules alongside the MHSA blocks; and **Convpass$_{share}$**, which adopts the redundancy-reduction design. Note that VPT, Adapter, LoRA, and Convpass only contain one type of PETL module, and the network architecture is the same for all tasks; while NOAH focuses on architecture search to combine other existing PETL modules, resulting in a dynamic network architecture.

4.1.3 Setup

We use a ViT-B/16 [8] supervisedly pretrained on ImageNet-21K [6] for all methods. The networks are finetuned for 100 epochs except for NOAH, which also trains a supernet for another 500 epochs. The hidden dimension h of Adapter, AdaptFormer, Convpass, and Convpass$_{attn}$, as well as the rank r of LoRA are all set to 8. The hidden dimension h of Convpass$_{attn}$ is 32. The prompt length l of VPT follows the best recipe in its original paper. The hyperparameter s of Convpass and AdaptFormer is roughly searched from {0.01, 0.1, 1, 10, 100}. In this setting, Adapter, AdaptFormer and Convpass$_{attn}$ have similar numbers of trainable parameters, while the Convpass's trainable parameters are slightly more than LoRA's but fewer than VPT's and NOAH's. Other hyperparameters are listed in Table 3.

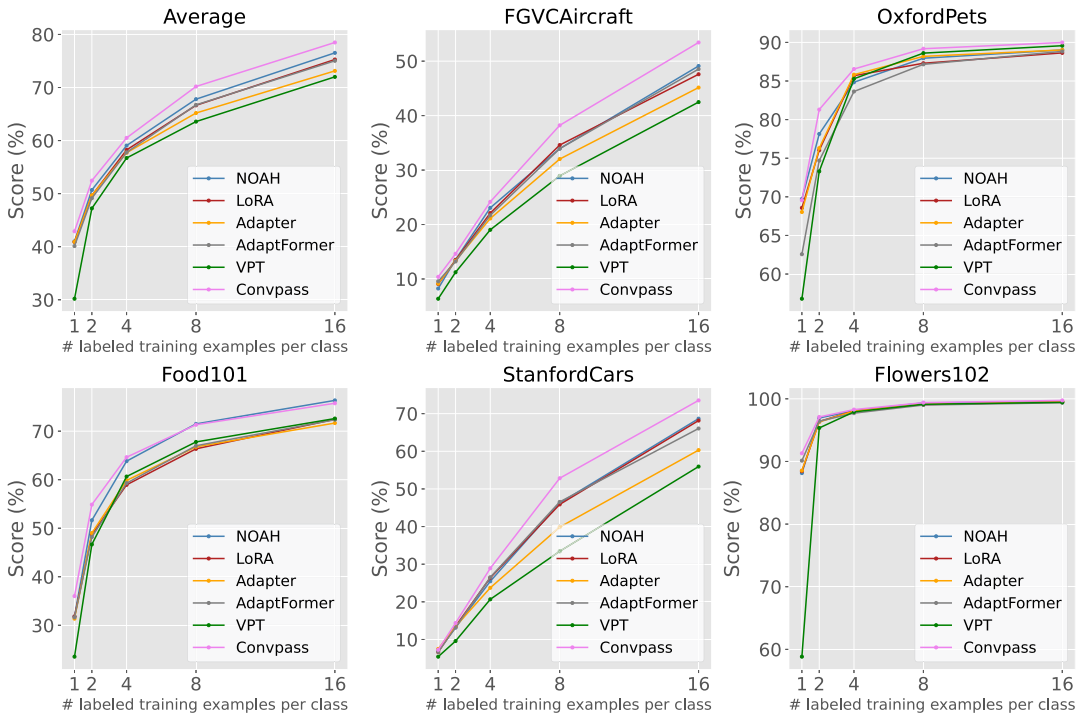


Figure 5: **Results of few-shot learning on five fine-grained visual recognition datasets.** Convpass outperforms other baselines on average results.

4.1.4 *Results*

As shown in Table 1, Convpass$_{attn}$ outperforms its counterpart Adapter and AdaptFormer on 16 and 10 out of the 19 tasks, while Convpass outperforms its counterparts LoRA and NOAH on 15 and 13 tasks, respectively. Although using fewer parameters, Convpass still performs better than VPT on 17 tasks. All the PETL methods are better than full finetuning overall. Because of the variety of tasks, no one method achieves SOTA on all tasks at once, but Convpass achieves the best average performance, 1.1% higher than the previous SOTA PETL methods, NOAH. It is worth noting that Convpass$_{attn}$ also has better average results than NOAH with only half as many parameters as NOAH. Moreover, since NOAH need to train an additional large supernet for architecture search, Convpass is also superior to NOAH in terms of training efficiency. Convpass$_{share}$ still performs on-par with NOAH, using 6× fewer parameters.

Figure 4 shows that Convpass has the best performance in all the three groups of VTAB, indicating that Convpass specializes in visual tasks from various domains. The superiority of Convpass is significant in the Natural and Structured groups. But in the Specialized group, Convpass does not remarkably outperform NOAH and AdaptFormer.

4.2 *Few-Shot Learning*

Few-shot learning is a common scenario when the data of downstream tasks is hard to obtain, and there are only a few training samples for each task that can be utilized.

4.2.1 *Datasets*

We use five fine-gained datasets to evaluate the performance of our methods on few-shot learning: **FGVC-Aircraft** [28], **Oxford-Pets** [30], **Food-101** [3], **Stanford Cars** [22], and **Oxford-Flowers102** [29]. We conduct experiments on 1, 2, 4, 8, and 16 shot settings. The results are averaged over three runs with different seeds. The experimental setup and baselines are the same as for VTAB-1K. Following [46], for training samples, we use color-jitter and RandAugmentation; for validation/test samples, we resize them to 256×256, crop them to 224×224 at the center, and then normalize them with ImageNet's mean and standard deviation.

Method	Source	Target			
	ImageNet	-V2	-S$_{ketch}$	-A	-R
ZS CLIP	66.73	60.83	46.15	47.77	73.96
LP CLIP	65.85	56.26	34.77	35.68	58.43
CoOp	71.51	64.20	47.99	49.71	75.21
CoCoOp	71.02	64.07	48.75	**50.63**	76.18
Tip-Adapter-F	73.41	65.39	48.58	49.23	77.54
Convpass$_{CLIP}$	**74.23**	**66.61**	**49.10**	49.27	**78.17**

Table 2: **Results of 16-shot ImageNet classification and domain generalization on CLIP.** We report top-1 accuracy. Convpass$_{CLIP}$ outperforms the baselines on source domain and three of the four target domains.

4.2.2 *Results*

As shown in Figure 5, the average results of Convpass are all higher than the other baselines across the five settings. On FGVC-Aircraft and Stanford Cars, the advantages of Convpass are highlighted. On simpler Oxford-Pets and Oxford-Flowers102, all the methods have similar performance, while Convpass is still in the lead. On Food-101, Convpass slightly underperforms NOAH in the 16-shot case, but the trend is reversed when the number of training data gets smaller. These results demonstrate that the introduced inductive bias of Convpass enhances ViT's capability to learn in the low-data regime.

	optimizer	batch size	learning rate	weight decay	# epochs	lr decay	# warm-up epochs
VTAB-1K	AdamW	64	1e-3	1e-4	100	cosine	10
Few-shot learning	AdamW	64	5e-3	1e-4	100	cosine	10
Domain generalization	AdamW	64	1e-5	0	50	cosine	0

Table 3: **Hyperparameters.**

4.3 Domain Generalization

Besides vision models, PETL has been studied in the field of vision-language models as well. Considering the outstanding domain generalization property of vision-language models, we also evaluate the performance of our method under domain shift when applied to vision-language models.

4.3.1 Datasets

In domain generalization experiments, the models are trained on the source domain, and tested on both the source and target domain. We use **ImageNet-1K** [6] as the source domain, where each class contains 16 training samples. The target domains include: **ImageNet-V2** [34], which is a new ImageNet test set collected with the original labelling protocol; **ImageNet-Sketch** [37], which consists of sketch images of the 1,000 ImageNet classes; **ImageNet-A** [13], which contains real-world adversarial samples of 200 of the ImageNet classes; **ImageNet-R** [12], which is composed of renditions of 200 ImageNet classes. Following [48], for training samples, we randomly resize and crop them to 224×224, and then implement random horizontal flip; for validation/test samples, we resize them to 224×224. All samples are finally normalized with ImageNet's mean and standard deviation.

4.3.2 Baselines

The CLIP [32] model consists of an image encoder and a text encoder, which are pretrained via contrastive learning on image-text pairs. Our method is compared with the following baselines: **Zero-Shot (ZS) CLIP** uses prompted label texts (e.g., "A photo of `<class name>`.") as the text encoder inputs, and classifies the images based on cosine similarity between image and text features ; **Linear Probe (LP) CLIP** discards the text encoder and learns a linear classification head for image encoder; **CoOp** [47] makes use of trainable vectors as prompts of labels; **CoCoOp** [48] learns a meta-net to generate prompts of labels from images; **Tip-Adapter-F** [45] caches features of training data to initialize an adapter after the image encoder. Note that CoOp, CoCoOp, and Tip-Adapter-F are PETL methods designed for CLIP specifically.

To apply our methods to CLIP, we make the following modifications. **First**, we insert Convpass modules into the image encoder only, while the text encoder stays unchanged. **Second**, we add a FC layer as classification head of the image encoder, whose bias is zero-initialized and whose weight is initialized with encoded prompted label texts of all classes (just as in ZS CLIP). Then, the text encoder is discarded, and only the Convpass modules and head are finetuned. We call this ported PETL method **Convpass**$_{CLIP}$.

4.3.3 Setup

In our experiments, all methods use a ViT-B/16 as the image encoder, and a BERT-like [7] model as the text encoder. For our methods, we train the Convpass modules and classification heads for 50 epochs. Other hyperparameters are listed in Table 3.

Model	Method	Avg.	Nat.	Spe.	Str.
ConvNeXt-B	Full	74.0	78.0	83.7	60.4
ConvNeXt-B	Linear	63.6	74.5	81.5	34.8
Swin-B	Full	75.0	79.2	86.2	59.7
Swin-B	Linear	62.6	73.5	80.8	33.5
Swin-B	VPT	71.6	76.8	84.5	53.4
Swin-B	Adaptformer	77.2	82.8	86.6	62.3
Swin-B	Convpass	**78.1**	**83.1**	**87.2**	**64.1**
ViT-B/16	Full	68.9	75.9	83.4	47.6
ViT-B/16	Linear	57.6	68.9	77.2	26.8
ViT-B/16	VPT	72.0	78.5	82.4	55.0
ViT-B/16	Adaptformer	74.7	80.5	84.9	58.8
ViT-B/16	Convpass	**76.6**	**81.7**	**85.3**	**62.7**

Table 4: **Results on VTAB-1K.** ConvNeXt-B and Swin-B have inherent inductive bias for vision, while ViT introduces such inductive bias via Convpass during finetuning. Avg.: Average, Nat.: Natural, Spe.: Specialized, Str.: Structured.

4.3.4 Results

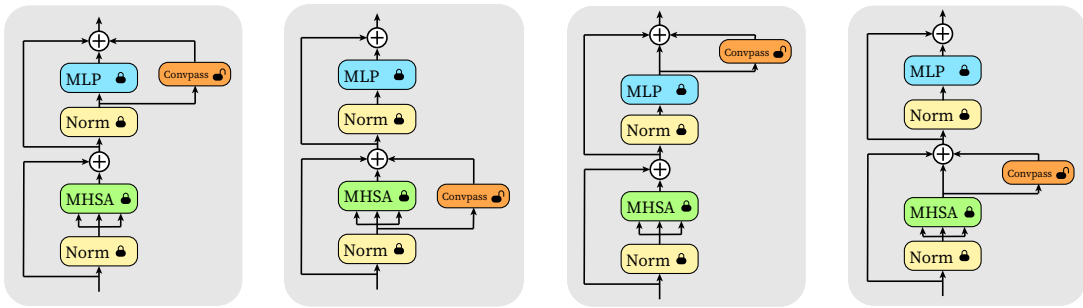
The results are shown in Table 2. Our method, though not designed for CLIP, still outperforms the baselines tailored for CLIP on the source domain. On three out of the four target domains, Convpass$_{CLIP}$ also achieves SOTA performance. On ImageNet-A, Convpass$_{CLIP}$ performs a bit poorly, which is probably because the ImageNet-A dataset is collected by selecting samples misclassified by ResNet. Since Convpass modules are ResNet-style blocks, they may be more easily misled by these samples as well. Overall, the results prove that Convpass$_{CLIP}$ is robust under domain shift.

4.4 Further Analyses

4.4.1 Comparison with Other Backbones

One of the motivations for designing Convpass is to introduce visual inductive bias to ViT during finetuning. However, since there are also ViT variants (e.g., Swin Transformer [26]) which have already incorporated visual inductive bias into their model designs, finetuning on these models can naturally benefit from such prior knowledge. Then a question arises: *Can the models that acquire inductive bias during finetuning outperform these models that have innately hard-coded inductive bias?*

We conduct comparisons among the three backbone models: **ViT-B/16**, **Swin-B** [26], and a SOTA CNN **ConvNeXt-B** [27]. All of them are pretrained on ImageNet-21K and have a similar size. As the results shown in Table 4, when using traditional transfer learning methods (Full and Linear), Swin-B and ConvNeXt-B perform significantly better than ViT-B/16 as expected, which indicates the pivotal role of visual inductive bias during finetuning. However, when equipped with Convpass, the average performance of ViT-B/16 overtakes fully finetuned Swin-B and ConvNeXt-B. These observations suggest that Convpass has the powerful capability to complement the missing inductive bias for downstream transfer tasks.



(a) Convpass$_{mlp}$ (b) Convpass$_{attn}$ (c) Seq-Convpass$_{mlp}$ (d) Seq-Convpass$_{attn}$

Figure 6: **Four ways to insert a Convpass module into ViT.**

Method	Avg.	Nat.	Spe.	Str.
Seq-Convpass$_{mlp}$	74.5	80.0	83.6	59.9
Seq-Convpass$_{attn}$	74.9	80.6	84.1	60.0
Convpass$_{mlp}$	75.4	80.3	84.5	61.2
Convpass$_{attn}$	**75.8**	**81.2**	**84.7**	**61.5**

Table 5: **Results on VTAB-1K.** We find that (i) parallel is superior to sequential, and (ii) alongside MHSA is superior to alongside MLP.

@MLP	@MHSA	Avg.	Nat.	Spe.	Str.
1×1	1×1	75.1	81.1	84.8	59.5
3×3	1×1	75.8	81.2	84.4	61.7
1×1	3×3	75.8	81.3	84.4	61.5
3×3	3×3	**76.6**	**81.7**	**85.3**	**62.7**

Table 6: **Results on VTAB-1K.** "1×1 @MLP" means the 3×3 convolutions in Convpass modules alongside the MLP blocks are replaced with 1×1 convolutions. We find that vision-oriented is superior to language-oriented.

Moreover, we also apply Convpass to Swin. Similarly, Convpass modules bypass the W-MHSA/SW-MHSA/MLP blocks of Swin. As shown in Table 4, the advantage of Convpass over full finetuning still holds on Swin, but VPT is no longer competitive. This observation demonstrates that Convpass is a reliable PETL method performing constantly well on various backbone networks. From the comparison between Swin and ViT we also find that the improvement made by Convpass diminishes on Swin. This is also expected because the demand for supplementing visual inductive on Swin is not as pressing as on ViT.

4.4.2 *Where to Place the Convpass Modules*

Our Convpass modules are parallel to the MHSA/MLP blocks, but there is another choice: insert the modules after the MHSA/MLP blocks in a sequential way like Adapter. To figure out what is the optimal way to place the Convpass modules, we consider four forms when only one Convpass module is inserted in each ViT layer , as illustrated in Figure 6. **Convpass$_{mlp}$** and **Convpass$_{attn}$** are parallel Convpass modules alongside the MLP and MHSA blocks, while **Seq-Convpass$_{mlp}$** and **Seq-Convpass$_{attn}$** follow the MLP and MHSA blocks, respectively.

As shown in Table 5, we evaluate these designs on VTAB-1K, and find the following. **First**, the parallel designs are better than their sequential counterparts. From Figure 2, we know that the sequential modules add longer paths to the model, which are relatively harder to optimize with a small amount of downstream data. On the contrary, the parallel Convpass serve as shortcuts for better gradient propagation, and introduce fully-convolutional ResNet-like paths that do not exist in sequential designs. **Second**, we also find that placing the Convpass modules beside/after MHSA blocks is better than beside/after MLP blocks. Since Convpass$_{attn}$ and Convpass$_{mlp}$ are the best two designs, our Convpass is composed of them, i.e., placing Convpass modules alongside both MHSA and MLP blocks in parallel.

4.4.3 *Vision-Oriented vs. Language-Oriented*

Finally, we conduct an ablation study on the vision-oriented idea. As shown in Table 6, we replace the 3×3 convolutions in Convpass modules alongside the MLP and/or MHSA with 1×1 convolutions, yielding four different designs. The bottom row is exactly Convpass. Replacing the 3×3 convolution in a Convpass module means the module will lose its capacity as a token mixer, degrading into a language-oriented adaptation module similar to Adapter.

The results show that, whether we replace the 3×3 convolutions of Convpass modules alongside MLP or alongside MHSA, the performance on Natural, Specialized, and Structured tasks will all degrade. If all 3×3 convolutions are replaced, the model will perform rather poorly on Structured. Since the Structured tasks are about obtaining the structure of a scene (e.g., object counting or 3D depth prediction), they fairly differ from the pretraining tasks (i.e., ImageNet classification) and require more modifications to the pretrained token mixer. Therefore, the Structured tasks are more complicated and the superiority of vision-oriented modules is highlighted. In summary, the language-oriented ablation models perform worse than the vision-oriented Convpass, supporting our standpoint.

5 Conclusion

In this paper, we point out that current PETL methods used in ViT lack inductive bias for visual tasks, which potentially degrades the performance on downstream finetuning. For this reason, we propose Convpass, a vision-oriented PETL method that employs trainable convolotional bypasses to adapt pretrained ViT to downstream tasks. Experimental results on VTAB-1K benchmark and few-shot learning show that Convpass outperforms other PETL methods and owns remarkable domain generalization property. Our simple but effective method reveals the importance of considering the characteristics of visual tasks when designing ViT-based PETL methods, which lights a promising direction for future work.

References

[1] L. J. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint*, arXiv: 1607.06450, 2016.

[2] H. Bao, L. Dong, and F. Wei. Beit: BERT pre-training of image transformers. In *ICLR*, 2022.

[3] L. Bossard, M. Guillaumin, and L. V. Gool. Food-101–mining discriminative components with random forests. In *ECCV*, 2014.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *arXiv preprint*, arXiv: 2005.14165, 2020.

[5] S. Chen, C. Ge, Z. Tong, J. Wang, Y. Song, J. Wang, and P. Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *arXiv preprint*, arXiv: 2205.13535, 2022.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[9] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig. Towards a unified view of parameter-efficient transfer learning. In *ICLR*, 2022.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[11] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. B. Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[12] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, D. Song, J. Steinhardt, and J. Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, 2021.

[13] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song. Natural adversarial examples. In *CVPR*, 2021.

[14] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for NLP. In *ICML*, 2019.

[15] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[16] M. Jia, L. Tang, B. Chen, C. Cardie, S. J. Belongie, B. Hariharan, and S. Lim. Visual prompt tuning. In *ECCV*, 2022.

[17] S. Jie and Z.-H. Deng. Fact: Factor-tuning for lightweight adaptation on vision transformer. In *Proceedings of AAAI*, 2023.

[18] S. Jie, H. Wang, and Z. Deng. Revisiting the parameter efficiency of adapters from the perspective of precision redundancy. In *Proceedings of ICCV*, 2023.

[19] S. Jie, Y. Tang, N. Ding, Z. Deng, K. Han, and Y. Wang. Memory-space visual prompting for efficient vision-language fine-tuning. In *Proceedings of ICML*, 2024.

[20] S. Jie, Y. Tang, J. Guo, Z. Deng, K. Han, and Y. Wang. Tocom: Altering inference cost of vision transformer without re-tuning. In *Proceedings of ECCV*, 2024.

[21] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby. Big transfer (bit): General visual representation learning. In *ECCV*, 2020.

[22] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *CVPR workshops*, 2013.

[23] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL/IJCNLP*, 2021.

[24] D. Lian, D. Zhou, J. Feng, and X. Wang. Scaling & shifting your features: A new baseline for efficient model tuning. In *Proceedings of NeurIPS*, 2022.

[25] X. Liu, K. Ji, Y. Fu, Z. Du, Z. Yang, and J. Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint*, arXiv: 2110.07602, 2021.

[26] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[27] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. In *CVPR*, 2022.

[28] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint*, arXiv: 1306.5151, 2013.

[29] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *CVPR*, 2006.

[30] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar. Cats and dogs. In *CVPR*, 2012.

[31] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *EACL*, 2021.

[32] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, Proceedings of Machine Learning Research, 2021.

[33] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21: 140:1–140:67, 2020.

[34] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? In *ICML*, 2019.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017.

[36] A. Veit, M. J. Wilber, and S. J. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *NIPS*, 2016.

[37] H. Wang, S. Ge, Z. C. Lipton, and E. P. Xing. Learning robust global representations by penalizing local predictive power. In *NeurIPS*, 2019.

[38] W. Wang, H. Bao, L. Dong, and F. Wei. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts. *arXiv preprint*, arXiv: 2111.02358, 2021.

[39] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang. Cvt: Introducing convolutions to vision transformers. In *ICCV*, 2021.

[40] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint*, arXiv: 2205.01917, 2022.

[41] E. B. Zaken, Y. Goldberg, and S. Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *ACL*, 2022.

[42] X. Zhai, J. Puigcerver, A. Kolesnikov, P. Ruyssen, C. Riquelme, M. Lucic, J. Djolonga, A. S. Pinto, M. Neumann, A. Dosovitskiy, L. Beyer, O. Bachem, M. Tschannen, M. Michalski, O. Bousquet, S. Gelly, and N. Houlsby. The visual task adaptation benchmark. *arXiv preprint*, arXiv: 1910.04867, 2019.

[43] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer. Scaling vision transformers. In *CVPR*, 2022.

[44] J. O. Zhang, A. Sax, A. R. Zamir, L. J. Guibas, and J. Malik. Side-tuning: A baseline for network adaptation via additive side networks. In A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, editors, *ECCV*, 2020.

[45] R. Zhang, R. Fang, P. Gao, W. Zhang, K. Li, J. Dai, Y. Qiao, and H. Li. Tip-adapter: Training-free adaption of clip for few-shot classification. In *ECCV*, 2022.

[46] Y. Zhang, K. Zhou, and Z. Liu. Neural prompt search. *arXiv preprint*, arXiv: 2206.04673, 2022.

[47] K. Zhou, J. Yang, C. C. Loy, and Z. Liu. Learning to prompt for vision-language models. *arXiv preprint*, arXiv: 2109.01134, 2021.

[48] K. Zhou, J. Yang, C. C. Loy, and Z. Liu. Conditional prompt learning for vision-language models. In *CVPR*, 2022.