# DTL: Disentangled Transfer Learning for Visual Recognition

**Minghao Fu, Ke Zhu, Jianxin Wu***

National Key Laboratory for Novel Software Technology, Nanjing University, China
School of Artificial Intelligence, Nanjing University, China
fumh@lamda.nju.edu.cn, zhuk@lamda.nju.edu.cn, wujx2001@gmail.com

## Abstract

When pre-trained models become rapidly larger, the cost of fine-tuning on downstream tasks steadily increases, too. To economically fine-tune these models, parameter-efficient transfer learning (PETL) is proposed, which only tunes a tiny subset of trainable parameters to efficiently learn quality representations. However, current PETL methods are facing the dilemma that during training the GPU memory footprint is *not* effectively reduced as trainable parameters. PETL will likely fail, too, if the full fine-tuning encounters the out-of-GPU-memory issue. This phenomenon happens because trainable parameters from these methods are generally entangled with the backbone, such that a lot of intermediate states have to be stored in GPU memory for gradient propagation. To alleviate this problem, we introduce Disentangled Transfer Learning (DTL), which disentangles the trainable parameters from the backbone using a lightweight Compact Side Network (CSN). By progressively extracting task-specific information with a few low-rank linear mappings and appropriately adding the information back to the backbone, CSN effectively realizes knowledge transfer in various downstream tasks. We conducted extensive experiments to validate the effectiveness of our method. The proposed method not only reduces a large amount of GPU memory usage and trainable parameters, but also outperforms existing PETL methods by a significant margin in accuracy, achieving new state-of-the-art on several standard benchmarks.

## Introduction

The pipeline of large-scale pre-training plus fine-tuning has been popularized in various domains (Devlin et al. 2018; Lewis et al. 2019; He et al. 2022; Caron et al. 2021). But traditional fine-tuning can be intractable due to GPU memory or time budget (He et al. 2022), since parameters of the entire large model have to be updated. Recently, parameter-efficient transfer learning (PETL) is proposed to update only a tiny subset of trainable parameters (Houlsby et al. 2019). Because of its efficacy and the ability to prevent over-fitting, numerous variants (Jia et al. 2022; Hu et al. 2022; Zhang, Zhou, and Liu 2022; Lian et al. 2022; Jie and Deng 2023) of PETL successively emerged.

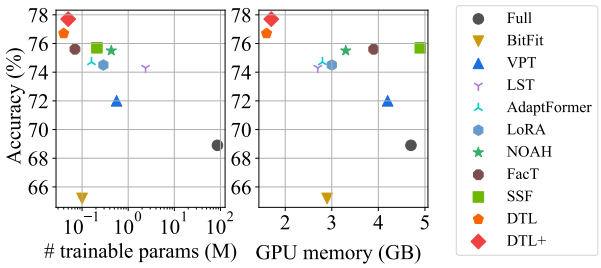


Figure 1: Top-1 accuracy on VTAB-1K (Zhai et al. 2019) vs. different numbers of trainable parameters and GPU memory footprint. Our DTL achieves the highest accuracy with the least trainable parameters and GPU memory usage.

Nevertheless, a huge decrease on trainable parameters does *not* necessarily mean an equivalent reduction in GPU memory usage: the percentage of saved GPU memory is still small (around 25%, cf. Fig. 1). Even the PETL pipeline may still fail if a large model cannot be fine-tuned due to GPU memory shortage. This drawback is critical and fundamental. Hence, it is critical to devise a new method that effectively reduces GPU memory usage and fully explores the utility of large-scale pre-trained models.

One common characteristic of PETL methods (Hu et al. 2022; Houlsby et al. 2019; Jia et al. 2022) is that *they closely entangle the small trainable modules with the huge frozen backbone*. As indicated by Sung, Cho, and Bansal (2022), for a specific network parameter to be correctly updated, the model has to cache related intermediate gradients from activation values. This entangled design makes the cache a considerable part of GPU memory footprint, and thus hinders large pre-trained models from being applied in various tasks.

To address this fundamental drawback, we propose Disentangled Transfer Learning (DTL), which *disentangles* the weights update from the backbone network by proposing a lightweight Compact Side Network (CSN). DTL not only greatly reduces GPU memory footage, but also achieves high accuracy in knowledge transfer (cf. Fig. 1).

As shown in Fig. 2, CSN composes of several low-rank linear mapping matrices to extract task-specific information, which is completely disentangled from the backbone. By in-

*J. Wu is the corresponding author.

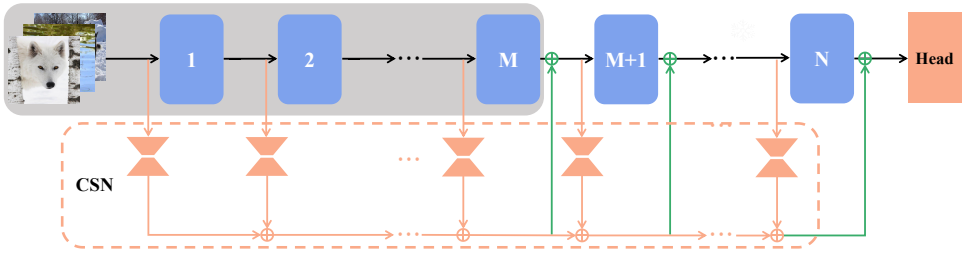


Figure 2: Illustration of DTL's network architecture for ViT (Dosovitskiy et al. 2021). Our Compact Side Network (CSN) with scarce trainable parameters is plugged *in parallel to* the backbone blocks. Specifically, before the forward calculation in each block, a low-rank linear mapping (Hu et al. 2022) is applied to the input features to aggregate task-specific side information (orange arrows). This side information is added back to the output of later backbone blocks (green arrows) for adapting backbone features to downstream tasks. During fine-tuning, only parameters of the CSN module and the task-specific classification head are updated (illustrated in orange). Best viewed in color.

jecting this information back to a few later backbone blocks, part of the intermediate features generated by pre-trained model are adaptively calibrated to make the features more discriminative for downstream tasks. We can also enhance DTL to DTL+, which inserts an additional global depth-wise separable convolution (DWConv) layer (Chollet 2017) to gather spatial information when injecting back from CSN to the backbone. DTL is very simple and compatible with various backbone architectures.

The output of early blocks in the backbone (covered by the gray region in Fig.2) is kept constant during fine-tuning, making it possible to reuse backbone features across *multiple downstream tasks* when the same input is provided.

We conducted extensive experiments to verify the effectiveness of the proposed DTL, which achieved superior top-1 evaluation accuracy with significantly less trainable parameters and GPU memory during fine-tuning compared to its traditional PETL counterparts. Our contributions can be summarized as follows:

- We analyze limitations of existing PETL methods from the perspective of GPU memory usage, which has a critical influence on the feasibility of fine-tuning.
- Motivated by our analysis, we propose DTL, a disentangled and simple framework for efficiently fine-tuning large-scale pre-trained models with significantly less trainable parameters and GPU memory usage.
- Extensive experiments are conducted to verify the effectiveness of DTL, which outperforms existing methods with a large margin.

## Related Work

PETL adapts a large pre-trained model to downstream tasks in a parameter-efficient fashion. Now we present some typical PETL methods in both vision and language communities.

BitFit (Ben Zaken, Goldberg, and Ravfogel 2022) fine-tunes all bias terms in the backbone network to partially adapt pre-trained models to downstream tasks. VPT (Jia et al. 2022) introduces prompt-tuning, which prepends learnable tokens $P \in \mathbb{R}^{l\times d}$ to patch tokens $X \in \mathbb{R}^{n\times d}$ as $X' = [P, X]$ to act as the input of a ViT block. Jia et al. (2022) propose two variants: 1) VPT-Shallow, which only inserts $P$ before the first block; and 2) VPT-Deep, where the input of every block is concatenated with a different $P$. During fine-tuning, only $P$ along with the classification head $W$ are learnable.

Adapter (Houlsby et al. 2019) fine-tunes text Transformers (Vaswani et al. 2017; Devlin et al. 2018) with a bottleneck architecture consisting of a down projection layer $W_{down} \in \mathbb{R}^{d\times d'}$ and an up projection layer $W_{up} \in \mathbb{R}^{d'\times d}$. It's inserted after the Multi-Head Self-Attention (MHSA) and Feed-Forward Network (FFN). The computation is formulated as $X' = X + \Theta(XW_{down})W_{up}$, where $\Theta$ is the activation function and $X$ is the output of MHSA or FFN. By setting $d' \ll d$, the number of trainable parameters ($\{W_{down}, W_{up}\}$) is limited. AdaptFormer (Chen et al. 2022) further attaches the bottleneck to FFN in a parallel form:

$$X' = X + \mathcal{FFN}(\mathcal{LN}(X)) + s \cdot \Theta(XW_{down})W_{up}\,, \quad (1)$$

where $X$ is the output of MHSA, $\mathcal{LN}$ is layer normalization (Ba, Kiros, and Hinton 2016) and $s$ is a scalar factor.

SSF (Lian et al. 2022) linearly transforms the intermediate features $X$ of the backbone with scale $\gamma \in \mathbb{R}^d$ and shift $\beta \in \mathbb{R}^d$, as $X' = \gamma \odot X + \beta$, in which $\odot$ denotes element-wise product and $X$ comes from the output of all MHSA, FFN and LN operations. Like other approaches, backbone parameters are frozen while additional parameters $\{\gamma,\beta\}$ are set to be learnable during fine-tuning.

LoRA (Hu et al. 2022) decomposes the update of weights matrix $W$ in a linear layer with $W' = W + \Delta W$. $\Delta W \in \mathbb{R}^{d\times d}$ is implemented by a low-rank approximation using two matrices $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times d}$, with $\Delta W = AB$ and $r \ll d$. Then the output after fine-tuning becomes

$$X' = XW + XAB\,. \quad (2)$$

By integrating $A$ and $B$ into both query ($W_q$) and value ($W_v$) mapping matrices in MHSA respectively, LoRA achieves superior results over previous works.

FacT (Jie and Deng 2023) boosts the efficiency of low-rank tuning using tensorization-decomposition to store the

update of trainable parameters, which contains two variants. The first one, termed as FacT-TT, decomposes $\Delta W$ as $\Delta W = s \cdot \Sigma \times_2 U^T \times_3 V^T$, where $U \in \mathbb{R}^{d \times r_1}$, $V \in \mathbb{R}^{d \times r_2}$, $\Sigma \in \mathbb{R}^{12L \times r_1 \times r_2}$ and $\times_i$ is mode-$i$ product. The other one, FacT-TK, further pushes the decomposition as $\Delta W = s \cdot C \times_1 P^T \times_2 U^T \times_3 V^T$, where $U \in \mathbb{R}^{d \times r_2}$, $V \in \mathbb{R}^{d \times r_3}$, $P \in \mathbb{R}^{12L \times r_1}$ and $C \in \mathbb{R}^{r_1 \times r_2 \times r_3}$, respectively. By setting $r_1, r_2, r_3 \ll d$, FacT is parameter-efficient.

NOAH (Zhang, Zhou, and Liu 2022) tries to free up researchers from manual architecture design. They propose to firstly train a supernet including Adapter, VPT and LoRA modules. After that, an evolutionary algorithm is performed to search the reduction dimensionality $d'$ in Adapter, prompt length $l$ in VPT and rank $r$ in LoRA under the constraint on the number of trainable parameters.

## Limitations of Current PETL Methods

Suppose there is an $N$ layer feed-forward network $y = f_N(f_{N-1}(...f_1(x)))$, where layer $i$ has a weight matrix $W_i$ and a bias term $b_i$. We denote $o_{i+1}$, $z_{i+1}$ as the output and pre-activation of layer $i$, respectively. Then, $o_{i+1} = \sigma(z_{i+1}) = \sigma(W_i o_i + b_i)$, where $\sigma$ is the activation function. Sung, Cho, and Bansal (2022) shows that the gradients back propagated from the loss $L$ to $W_i$ and $b_i$ are

$$\begin{aligned} \frac{\partial L}{\partial W_i} &= \frac{\partial L}{\partial o_{i+1}} \sigma_i' o_i \,, \\ \frac{\partial L}{\partial b_i} &= \frac{\partial L}{\partial o_{i+1}} \sigma_i' \,, \end{aligned} \tag{3}$$

where $\sigma_i'$ is the abbreviation of $\partial o_{i+1} / \partial z_{i+1}$. Furthermore, the term $\partial L / \partial o_{i+1}$ can be recursively expressed as

$$\frac{\partial L}{\partial o_{i+1}} = \frac{\partial L}{\partial o_{i+2}} \frac{\partial o_{i+2}}{\partial z_{i+2}} \frac{\partial z_{i+2}}{\partial o_{i+1}} = \frac{\partial L}{\partial o_{i+2}} \sigma_{i+1}' W_{i+1} \,. \tag{4}$$

To correctly calculate the gradients, except for parameters from the model (in this case, $W_i$ and $b_i$), *all corresponding* $\{\sigma_i'\}$ *in the chain rule have to be cached during fine-tuning*, which dominates the GPU memory usage.

We have introduced several representative PETL methods in the related work section, and *all* these methods closely *entangle* the trainable parameters with the backbone, which hardly reduces the GPU memory usage in caching $\{\sigma_i'\}$. This property shows that the GPU memory footprint cannot be effectively reduced compared to a full fine-tuning, even though the number of trainable parameters is very small.

To solve this fundamental difficulty, we propose a new learning paradigm called Disentangled Transfer Learning (DTL). The central idea of DTL is to *disentangle* the weight updating of the small extra modules from the the backbone network (cf. Fig. 2). Therefore, the relevant $\sigma_i'$ stored for back propagation can be drastically reduced (cf. Eq. 3 and 4). In this way, DTL successfully pushes the limits of current PETL further from not only being parameter-efficient but also reducing the necessary GPU memory size in fine-tuning large-scale pre-trained models.

## Method

We propose a *disentangled*, *simple* and *effective* approach to fine-tune large-scale pre-trained models properly. In order to trade off the recognition accuracy and architectural complexity in different environments, we introduce two variants of our method, termed as DTL and DTL+.

### DTL: Simplicity Matters

We first show the simplest version of our solution. In Fig. 2 we illustrate the pipeline of the proposed architecture for the ViT (Dosovitskiy et al. 2021) backbone, which is mainly built up with a *Compact Side Network* (CSN). CSN is plugged into the backbone for information aggregation and feature adaptation. Note that the proposed method is compatible with other types of backbones, which will be discussed soon.

Given a ViT backbone containing $N$ blocks, the forward calculation can be formulated as $z = b_N(b_{N-1}(...b_1(x)))$, where $b_i$ is the $i$-th block, $x \in \mathbb{R}^{(n+1) \times d}$ is the input tokens (patch tokens plus one *cls* token) and $z \in \mathbb{R}^{(n+1) \times d}$ is the output tokens, respectively. Denote $z_{i+1}$ as the output of $b_i$, hence $z_{i+1} = b_i(z_i)$ and $z_1 = x$. Our CSN composes of $N$ low-rank linear transformation matrices (Hu et al. 2022), with each being plugged into one block to extract task-specific information. Denote $w_i = a_i c_i \in \mathbb{R}^{d \times d}$ as the weight matrix accounting for the $i$-th block, with $a_i \in \mathbb{R}^{d \times d'}$, $c_i \in \mathbb{R}^{d' \times d}$ and $d' \ll d$, CSN progressively gathers information from each block as

$$h_{i+1} = h_i + z_i w_i \,, \tag{5}$$

$$z_{i+1} = b_i(z_i) \,, \tag{6}$$

where $h_{i+1}$ is the output of the $i$-th layer of CSN ($h_1 = 0$). After that, starting from the $M$-th block, the aggregated task-specific information $h_{i+1}$ is used to adapt $z_{i+1}$ to downstream tasks by adding it back to $z_{i+1}$. Hence, when $i \geq M$,

$$z'_{i+1} = z_{i+1} + \theta(h_{i+1}) \,, \tag{7}$$

where $z'_{i+1}$ is the adapted output of $b_i$ and $\theta$ is the Swish activation (Ramachandran, Zoph, and Le 2017), where $\theta(x) = \frac{x}{1+e^{-\beta x}}$. To prevent $z'_{i+1}$ from drastically shifting away from $z_{i+1}$ at the beginning of fine-tuning, $a_i$ is initialized following a uniform distribution and $c_i$ is zero-initialized. To sum up, the output from the $i$-th block is

$$z'_{i+1} = \begin{cases} z_{i+1} + \theta(h_{i+1}) & \text{if } i \geq M \\ z_{i+1} & \text{otherwise} \,. \end{cases} \tag{8}$$

We find that a small $d'$ (2 or 4) performs fairly well, which suggests high redundancy in the backbone features. Therefore, in addition to keep $d'$ small, we use a large $\beta$ (100) in Swish (i.e., $\theta$) to further reduce the redundancy. Consequently, about half of $\theta(h_{i+1})$ is close to zero.

### DTL+ : Effectiveness Matters

To further boost the effectiveness of the proposed method, we append an additional global depthwise separable convolution (DWConv) layer (Chollet 2017) $g$ to each side layer after $\theta$ is applied. The formulation of DTL+ is

$$z'_{i+1} = \begin{cases} z_{i+1} + g(\theta(h_{i+1})) & \text{if } i \geq M \\ z_{i+1} & \text{otherwise} \,. \end{cases} \tag{9}$$

The stride of $g$ is set to 1 and zero-padding is used to ensure that $g$ does not change feature size. Note that *$g$ is shared across different CSN layers*, so that the number of trainable parameters in $g$ is small compared to the initial CSN, and the whole CSN module is still lightweight. The introduction of $g$ makes spatial information properly processed by our CSN module. With this operation, it's easier for the model to recognize new categories.

## Advantages

The proposed approach has some significant advantages, which we discuss explicitly.

**Disentangled.** As shown in Fig. 2, the proposed CSN is a plugin mostly detached from the backbone, which interacts with the backbone in a plug-and-play manner. This characteristic makes our method easy to implement, and is *compatible with almost all backbone networks*. Modern deep neural networks are mostly divided into several intermediate stages and the feature dimensionality within one stage is the same. By re-initializing the hidden state of CSN $h_i$ to 0 at the beginning of each stage, our method can be easily transferred to different backbone architectures.

From the perspective of GPU memory usage, in previous methods weight update is directly entangled with the backbone. As we have analyzed before, although the number of trainable parameters is small, they still require a lot of GPU memory to cache many $\{\sigma_i'\}$ for gradient propagation. Our method alleviates this issue by 1) separating the forward pass of the backbone from CSN; and 2) only entangling them at late stages ($i \geq M$). Within our framework, *no gradients are back propagated to the first $M$ blocks in the backbone* (the gray region in Fig. 2). Hence, the number of cached $\{\sigma_i'\}$ is drastically reduced in CSN, resulting in a highly efficient way to realize GPU memory reduction.

Finally, we discuss another advantage drawn from our disentangled architecture: the possibility of feature reuse. Consider a scenario where we need to perform different tasks on one input image (e.g., simultaneously predict age and gender for a human). We have several fine-tuned models, and in previous methods the intermediate features $z_{i+1}$ generated by these fine-tuned models are *different to each other*. In other words, there is no way to share computation in the backbone across different tasks. Therefore a standard process is learning a group of task-specific parameters for each task (cf. Table 1) and conducting each task individually.

Conversely, as shown in the gray region of Fig. 2, in our DTL the intermediate features before block $M$ remain the same after fine-tuning, such that we can share part of the backbone computation between different tasks.

We take the 19 datasets in VTAB-1K (Zhai et al. 2019) to illustrate the above-described situation. We fine-tune to obtain 19 models, and assume that we need to get all 19 classification results for the one input image using all these 19 models. The goal is to check how much speedup can be achieved during inference. Firstly, we feed the same input image into 19 different models after fine-tuning with LoRA (Hu et al. 2022), which acts as the baseline. Then we implement our method to simultaneously conduct 19 tasks but with the backbone feature shared in the first 6 blocks (by setting $M = 7$). Experimental results show that approximately *45% inference latency is saved during inference*.

| Method | Source | #unit |
|---|---|---|
| LoRA | low-rank matrices in $W_q$, $W_v$ | 24 |
| NOAH | low-rank matrices, bottlenecks, prompts | 36 |
| FacT | decomposed tensors | 144 |
| SSF | pairs of $\gamma$, $\beta$ | 148 |
| DTL | low-rank matrices | 12 |
| DTL+ | low-rank matrices, DWConv | 13 |

Table 1: Statistics of number of minimal structural units in different methods. "Source" denotes the types of minimal structural units. "#unit" denotes the number of minimal structural units in the backbone.

**Simple**. Since the CSN is disentangled from the backbone network, our method naturally shows higher simplicity compared to previous methods. Since all PETL methods add various types of structural units as extra trainable parameters, to verify the simplicity of our DTL in more detail, we compare the number of such minimal structural units of our method with previous methods in Table 1.

In this context, the phrase *minimal structural unit* means the atomic modules inserted into the backbone network. For example, in LoRA (Hu et al. 2022), one minimal structural unit comprises of a pair of $A$ and $B$ matrices to constitute $\Delta W$ (cf. Eq. 2). Since it inserts $\Delta W$ into both $W_q$ and $W_v$ for MHSA in every Transformer block, the total number of these units is 24. It is similarly defined for other methods as well, which include: 1) pairs of $\gamma$ and $\beta$ in SSF; 2) the modules to be searched in supernet and maintained in subnet in NOAH; 3) decomposed tensors in FacT; 4) pairs of matrices $a_i$ and $c_i$ in our DTL; 5) the additional global DWConv layer in DTL+. As shown in Table 1, the proposed method requires much fewer minimal structural units compared to existing methods.

We notice that a previous work LST (Sung, Cho, and Bansal 2022) also has a side network design. However, their architecture is very complicated and requires sophisticated techniques (Li et al. 2017) to initialize, resulting in the existence of large number of trainable parameters as shown in Table 2 (about 50$\times$ compared to ours). As analyzed before, we reduce the fine-tuning redundancy by setting $d'$ to a very small value (2 or 4), which is far less than previous counterparts (e.g., 8 in LoRA and Adapter). This choice makes our DTL not only simple in structure, but also contains much fewer trainable parameters than other methods.

**Effective.** We conducted extensive experiments to verify the effectiveness of the proposed method. The results demonstrate that our method shows superior recognition accuracy across multiple architectures, achieving new state-of-the-art on several standard benchmarks.

# Experiments

We conducted thorough experiments to evaluate the proposed method. First, we present results on the VTAB-1K (Zhai et al. 2019) benchmark with two prevalent backbones, ViT-B/16 (Dosovitskiy et al. 2021) and Swin-B (Liu

| | #param (M) | GPU mem (GB) | Natural | | | | | | | Specialized | | | | Structured | | | | | | | | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 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 | |
| *Traditional Fine-Tuning* | | | | | | | | | | | | | | | | | | | | | | |
| Full | 85.8 | 4.7 | 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 | 0.6 | 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* | | | | | | | | | | | | | | | | | | | | | | |
| BitFit | 0.10 | 2.9 | 72.8 | 87.0 | 59.2 | 97.5 | 85.3 | 59.9 | 51.4 | 78.7 | 91.6 | 72.9 | 69.8 | 61.5 | 55.6 | 32.4 | 55.9 | 66.6 | 40.0 | 15.7 | 25.1 | 65.2 |
| VPT | 0.56 | 4.2 | 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 |
| LST | 2.38 | 2.7 | 59.5 | 91.5 | 69.0 | 99.2 | 89.9 | 79.5 | 54.6 | 86.9 | 95.9 | 85.3 | 74.1 | 81.8 | 61.8 | 52.2 | 81.0 | 71.7 | 49.5 | 33.7 | 45.2 | 74.3 |
| LoRA | 0.29 | 3.0 | 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 |
| AdaptFormer | 0.16 | 2.8 | 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 |
| NOAH | 0.43 | 3.3 | 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 |
| FacT | 0.07 | 3.9 | 70.6 | 90.6 | 70.8 | 99.1 | 90.7 | 88.6 | 54.1 | 84.8 | 96.2 | 84.5 | 75.7 | 82.6 | 68.2 | 49.8 | 80.7 | 80.8 | 47.4 | 33.2 | 43.0 | 75.6 |
| SSF | 0.21 | 4.9 | 69.0 | 92.6 | **75.1** | 99.4 | 91.8 | 90.2 | 52.9 | 87.4 | 95.9 | 87.4 | 75.5 | 75.9 | 62.3 | **53.3** | 80.6 | 77.3 | 54.9 | 29.5 | 37.9 | 75.7 |
| DTL | **0.04** | **1.6** | 69.6 | 94.8 | 71.3 | 99.3 | 91.3 | 83.3 | 56.2 | 87.1 | 96.2 | 86.1 | 75.0 | 82.8 | 64.2 | 48.8 | 81.9 | 93.9 | 53.9 | 34.2 | 47.1 | 76.7 |
| DTL+ | 0.05 | 1.7 | 70.4 | **95.1** | 71.5 | **99.4** | **91.8** | 87.5 | 56.8 | 87.7 | 96.6 | 86.9 | 74.7 | 81.6 | 65.1 | 51.3 | **82.3** | **97.2** | **54.9** | **36.0** | 49.3 | 77.7 |
| DTL+* | 0.05 | 3.1 | **74.1** | 94.8 | 71.8 | 99.4 | 91.7 | **90.4** | **57.2** | **87.9** | **96.7** | **87.5** | 74.8 | 81.9 | 64.7 | 51.5 | 81.9 | 93.9 | 54.0 | 35.6 | **50.3** | **77.9** |

Table 2: Results on the VTAB-1K benchmark with ViT-B/16 as the backbone. "#param" denotes the number of trainable parameters. "GPU mem" specifies the peak GPU memory footprint when fine-tuning with batch size 32. "Average" is the group-wise average accuracy over three groups. The best results among PETL methods are in bold face.

et al. 2021). Then we verify the generalization ability of our method on few-shot learning and domain generalization. Finally, ablation studies are conducted for further analysis.

## Implementation Details

Following previous work (Lian et al. 2022; Jie and Deng 2023), we take AdamW (Loshchilov and Hutter 2019) with cosine learning rate schedule as the optimizer. $\beta$ in Swish is fixed to 100. All pre-trained models are fine-tuned by 100 epochs with batch size 32. The rank $d'$ of low-rank linear mappings in CSN is 2 for ViT and 4 for Swin-B. We set $M$ (cf. Eq. 8-9) of DTL and DTL+ as 7 for the ViT backbone, which means half of the later blocks' output is calibrated by adding back the output from CSN. It is similarly defined for Swin-B with half of the layers adapted as well. Note that unlike previous methods (Zhang, Zhou, and Liu 2022; Lian et al. 2022), except for the standard data augmentation, we do *not* use any additional tricks such as mixup (Zhang et al. 2018), cutmix (Yun et al. 2019) or label smoothing (Szegedy et al. 2016). More details are available at https://www.lamda.nju.edu.cn/fumh/files/DTL/DTL_appendix.pdf.

## Experiments on VTAB-1K

**Datasets.** VTAB-1K was introduced by Zhai et al. (2019) to evaluate the generalization ability of representation learning approaches. It contains diverse images from 19 different datasets, grouped as 1) *Natural* images captured by standard cameras; 2) *Specialized* images captured by specialist equipment; and 3) *Structured* images generated in simulated environments. They vary in task-specific objectives (e.g., classic visual recognition, object counting or depth prediction) and there are only 1,000 images in each dataset for training. It is a challenging benchmark to evaluate PETL methods. We report the top-1 recognition accuracy on the test set.

**Baseline methods.** First, two traditional fine-tuning techniques are included in all experiments. One is 'Full', which fine-tunes the entire pre-trained model. The other is 'Linear', which only fine-tunes task-specific classification head. Second, we choose BitFit (Ben Zaken, Goldberg, and Ravfogel 2022), VPT (Jia et al. 2022), LST (Sung, Cho, and Bansal 2022), AdaptFormer (Chen et al. 2022), LoRA (Hu et al. 2022), NOHA (Zhang, Zhou, and Liu 2022), FacT (Jie and Deng 2023) and SSF (Lian et al. 2022) as PETL baselines. We follow the setting in (Zhang, Zhou, and Liu 2022; Lian et al. 2022) to report the results for a fair comparison.

In addition to DTL and DTL+ ($M = 7$), we further extend DTL+ where all of the blocks are adapted (i.e., $M = 1$, denoted as 'DTL+*').

**Main results.** Results on ViT-B/16 are shown in Table 2. Our DTL shows a 1.0% gain on average accuracy compared to the previous state-of-the-art method SSF. By integrating a global DWConv, DTL+ further increases the average improvement to 2.0%. Specifically, DTL+ reaches the best top-1 accuracy on 11 out of 19 datasets, where the improvements compared to SSF range from 0.3% to 19.9%. Even if the dataset 'dSpr-Loc' with the most significant gain of 19.9% is removed, DTL+ is still far ahead on average accuracy of the remaining 18 datasets and outperforms SSF by 1.3%.

DTL only introduces 0.04M trainable parameters, which is 43% less compared to FacT. DTL+ specifies a few more trainable parameters (+0.01M) because of the introduction of a shared DWConv, but is still significantly less than pre-

| Method | #p | #m | Nat. | Spe. | Str. | Avg. |
|---|---|---|---|---|---|---|
| Full | 86.7 | 6.1 | 79.2 | 86.2 | 59.7 | 75.0 |
| Linear | 0 | 0.9 | 73.5 | 80.8 | 33.5 | 62.6 |
| BitFit | 0.20 | 3.7 | 74.2 | 80.1 | 42.4 | 65.6 |
| VPT | 0.16 | 4.6 | 76.8 | 84.5 | 53.4 | 71.6 |
| FacT | 0.14 | 5.6 | 83.1 | 86.9 | 62.1 | 77.4 |
| DTL | **0.09** | **1.5** | 82.4 | **87.0** | 64.2 | 77.9 |
| DTL+ | 0.13 | 1.6 | 82.4 | 86.8 | **66.0** | 78.4 |
| DTL+* | 0.14 | 4.0 | **83.2** | 87.0 | 65.7 | **78.6** |

Table 3: Results on VTAB-1K with Swin-B backbone. '#p' is the number of trainable parameters. '#m' is peak GPU memory footprint in fine-tuning. Nat./Spe./Str./Avg. are the results in three VTAB groups and their group-wise average.

| Method | **Source** | **Target** | | | |
|---|---|---|---|---|---|
| | ImageNet | -Sketch | -V2 | -A | -R |
| Adapter | 70.5 | 16.4 | 59.1 | 5.5 | 22.1 |
| VPT | 70.5 | 18.3 | 58.0 | 4.6 | 23.2 |
| LoRA | 70.8 | 20.0 | 59.3 | 6.9 | 23.3 |
| NOAH | 71.5 | 24.8 | 66.1 | 11.9 | 28.5 |
| DTL | 78.3 | 35.4 | 67.8 | 14.0 | 34.4 |
| DTL+ | **78.7** | **35.7** | **67.8** | **14.2** | **34.4** |

Table 4: Top-1 accuracy on domain generalization experiments with ViT-B/16 as the backbone. Our method shows significant gains w.r.t baseline methods.

vious PETL methods. Moreover, DTL and DTL+ consume 1.6GB and 1.7GB GPU memory during fine-tuning, respectively, which is far less than other PETL baselines. Compared to full fine-tuning, the GPU memory saving rate is about 65% on average (or saving roughly two thirds).

Another interesting observation is DTL+ and DTL+* show different distributions on accuracy improvements between three groups. For the 'Natural' group with smaller domain discrepancy between pre-trained models and downstream tasks, DTL+* outperforms DTL+ by 1%. For the 'Structured' group with a large domain gap, DTL+ surpasses DTL+* by 0.5% instead. This observation also appears similarly in Table 3 on Swin-B. We thus conjecture that since DTL+* has larger capacity than DTL+, it is prone to overfitting when facing large domain gaps.

We observe that DTL+* shows more improvements in top-1 accuracy compared to DTL+, which indicates that the feature adaptation in early blocks within backbone is, in general, useful for transfer learning. However, the GPU memory usage in this case is increased to 3.1 GB, yet with only 0.2% accuracy gain compared to DTL+, implying the limited cost-efficiency of adapting early features.

As presented in Table 3, the performance of DTL and DTL+ on Swin-B shows similar trends with ViT. DTL+ achieves new state-of-the-art, outperforming FacT with a significant margin of 1% on average accuracy. DTL keeps the least trainable parameters and GPU memory footprint. Compared to full fine-tuning, DTL drastically saves GPU memory usage by 75%.

## Experiments on Few-shot Learning

**Datasets.** Now we further evaluate on five fine-grained few-shot learning benchmark: Aircraft (Maji et al. 2013), Pets (Parkhi et al. 2012), Food-101 (Bossard, Guillaumin, and Van Gool 2014), Cars (Krause et al. 2013) and Flowers102 (Nilsback and Zisserman 2008). Following Jie and Deng (2023), we fine-tune the pre-trained model with training set containing $\{1, 2, 4, 8, 16\}$-shot per class and report the average accuracy on test set over 3 seeds.

**Main results.** As illustrated in Fig. 3, the proposed DTL and DTL+ outperform all baseline PETL methods in all cases. Furthermore, we observe that the average improvements of DTL+ compared to previous state-of-the-art FacT across different shots are gradually decreased from 4.7% in 1-shot to 0.8% in 16-shot, which reveals that our method is consistently effective, especially in low-data regimes.

## Experiments on Domain Generalization

**Datasets.** We follow Zhang, Zhou, and Liu (2022) to conduct experiments on domain generalization to evaluate the robustness of our method when domain shift (Zhou et al. 2023) is inevitable. In this scenario, the training set to fine-tune the pre-trained ViT-B/16 model is sampled from the original training set of ImageNet-1K, with each class containing 16 shot of images. Apart from the validation set of ImageNet-1K, the model is evaluated on 4 datasets, which are 1) ImageNet-Sketch (Wang et al. 2019) composed of sketch images sharing the same label space with ImageNet-1K, 2) ImageNet-V2 (Recht et al. 2019) collected from different sources compared with ImageNet-1K, 3) ImageNet-A (Hendrycks et al. 2021b) consisting of adversarial examples, and 4) ImageNet-R (Hendrycks et al. 2021a) containing various artistic renditions of ImageNet-1K. The reported accuracy is average by 3 different random seeds.

**Main results.** The results of domain generalization experiments are shown in Table 4. We observe that compared to previous state-of-the-art method NOAH, DTL and DTL+ achieve impressive gains in evaluation accuracy, especially on ImageNet, ImageNet-Sketch and ImageNet-R, where the average improvement is up to about 8%. These comparisons show excellent robustness of DTL and DTL+ for alleviating the domain shift problem and well demonstrate the effectiveness of the proposed method together with previous experiments.

## Ablation Studies

**Sensitivity to $M$.** In DTL and DTL+, the beginning index ($M$) of blocks in the backbone to add back the output of CSN for feature adaptation is critical for final performance. In Fig. 4 we plot the curve of accuracy and GPU memory footprint by varying $M$. There is a clear trend that the number of GPU memory usage decreases almost linearly as $M$ becomes larger. By decreasing $M$ from 12 to 11, the recognition accuracy is significantly boosted to 76.7% from 75.9%, and the improvements gradually saturate when $M < 6$, implying the feasibility and effectiveness of feature sharing as we described in the methods section. Finally, the

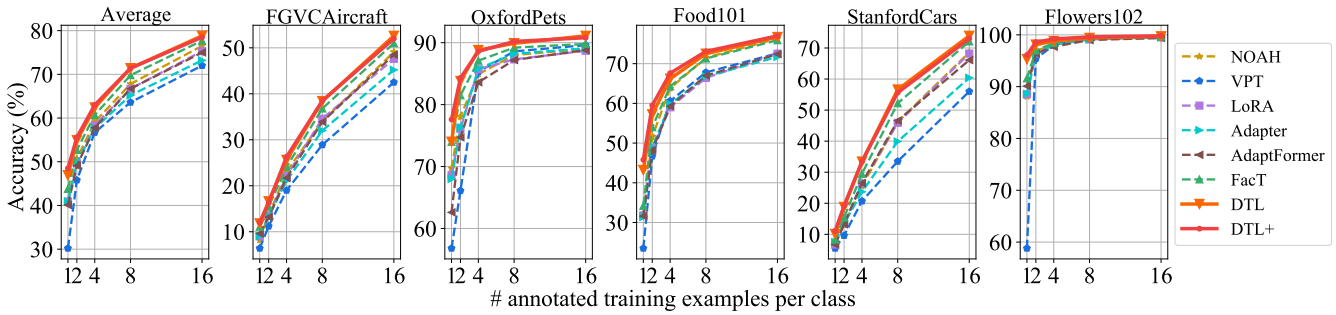


Figure 3: Top-1 accuracy on fine-grained few-shot benchmark with ViT-B/16 as the backbone. Best viewed in color. Note that our approach with less trainable parameters and GPU memory footprint outperforms all baseline methods.

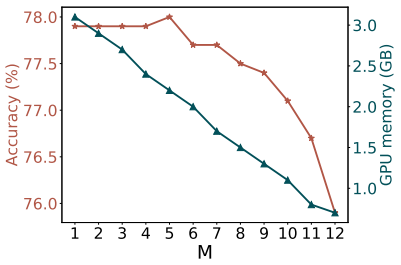


Figure 4: Top-1 accuracy and peak GPU memory footprint under various $M$ in Eq. 9. Our method is consistently effective across different $M$.

| $d'$ | Swish ($\theta$) | DWConv ($g$) | Avg. |
|---|---|---|---|
| 2 | | | 76.0 |
| 2 | ✓ | | 76.7 |
| 2 | ✓ | ✓ | 77.7 |
| 4 | ✓ | ✓ | 77.6 |
| 1 | ✓ | ✓ | 77.2 |

Table 5: Ablation results by varying different architectural choices, where the second and third lines denote default DTL and DTL+, respectively.

range of accuracy across different $M$ is 2.1%, indicating that the recognition accuracy is not sensitive to the exact value of $M$, so by default we set $M = 7$ to pursue the best trade-off between effectiveness and efficiency.

**Modular ablation.** In Table 5, we provide ablation results on the VTAB-1K benchmark from ViT-B/16. The first line, where the layer-wise output of CSN with rank ($d' = 2$) is directly added back to backbone (i.e, $z'_{i+1} = z_{i+1} + h_{i+1}$ when $i \geq M$), is the baseline. It achieves a 76.0% average accuracy. By progressively integrating the Swish activation function $\theta$ in DTL and a global DWConv $g$ in DTL+, the accuracy is consistently improved. For DTL+, a higher ($d' = 4$) or lower ($d' = 1$) rank both make the accuracy worse than the default ($d' = 2$). Interestingly even when the trainable parameters of CSN are extremely limited with $d' = 1$, our DTL+ is still more effective than previous PETL methods.

| Methods | Throughput (imgs/sec) | | |
|---|---|---|---|
| | bs=1 | bs=4 | bs=16 |
| Full | 161 | 636 | 952 |
| LST | 71 | 279 | 729 |
| NOAH | 79 | 306 | 798 |
| AdaptFormer | 108 | 436 | 876 |
| DTL+ | 120 | 469 | 877 |
| DTL | 131 | 528 | 892 |

Table 6: Throughput (number of images processed per second with ViT-B/16 as the backbone) measured on a single NVIDIA 3090 GPU with mixed precision inference.

**Inference efficiency.** We further study the efficiency of our method during inference by comparing the throughput with some baselines. As illustrated in Table 6, thanks to its simplicity, the empirical throughput of DTL+ is consistently higher than previous PETL counterparts. The simplest version of our method, DTL, boosts the inference efficiency a step further and significantly shows more speedup compared to traditional PETL methods.

## Conclusions and Limitations

In this paper, we proposed Disentangled Transfer Learning, a new paradigm for fine-tuning large-scale pre-trained models. To trade off the efficiency and effectiveness, we designed two variants, DTL and DTL+. The most important property of DTL is, by disentangling weights update of trainable parameters from the backbone, it drastically reduces the GPU memory footprint required during fine-tuning. At the same time, the proposed method contains less trainable parameters and achieves competitive or even better accuracy compared to traditional PETL methods. Extensive experiments on several standard benchmarks plus ablations clearly show that our method is not only effective but also efficient for fine-tuning, indicating its great potential for practical usage.

An obvious limitation of DTL is that its granularity of interaction between the backbone and the trainable modules is very coarse. This is caused by the disentangled design. In the future, a better trade-off may be made between the two desired properties: disentanglement and interaction.

## Acknowledgments

This research was partly supported by the National Natural Science Foundation of China under Grant 62276123 and Grant 61921006.

## References

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

Ben Zaken, E.; Goldberg, Y.; and Ravfogel, S. 2022. BitFit: Simple parameter-efficient fine-tuning for Transformer-based masked language-models. In *60th Annual Meeting of the Association for Computational Linguistics*, 1–9.

Bossard, L.; Guillaumin, M.; and Van Gool, L. 2014. Food-101 – Mining discriminative components with Random Forests. In *European Conference on Computer Vision*, volume 8694 of *LNIP*, 446–461. Springer.

Caron, M.; Touvron, H.; Misra, I.; Jegou, H.; Mairal, J.; Bojanowski, P.; and Joulin, A. 2021. Emerging properties in self-supervised Vision Transformers. In *IEEE/CVF International Conference on Computer Vision*, 9630–9640.

Chen, S.; GE, C.; Tong, Z.; Wang, J.; Song, Y.; Wang, J.; and Luo, P. 2022. AdaptFormer: Adapting Vision Transformers for scalable visual recognition. In *Advances in Neural Information Processing Systems*, 16664–16678.

Chollet, F. 2017. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1800–1807.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of deep bidirectional Transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 1–21.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollar, P.; and Girshick, R. 2022. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15979–15988.

Hendrycks, D.; Basart, S.; Mu, N.; Kadavath, S.; Wang, F.; Dorundo, E.; Desai, R.; Zhu, T.; Parajuli, S.; Guo, M.; Song, D.; Steinhardt, J.; and Gilmer, J. 2021a. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *IEEE/CVF International Conference on Computer Vision*, 8320–8329.

Hendrycks, D.; Zhao, K.; Basart, S.; Steinhardt, J.; and Song, D. 2021b. Natural adversarial examples. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15257–15266.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2790–2799.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 1–13.

Jia, M.; Tang, L.; Chen, B.-C.; Cardie, C.; Belongie, S.; Hariharan, B.; and Lim, S.-N. 2022. Visual prompt tuning. In *European Conference on Computer Vision*, volume 13693 of *LNCS*, 709–727. Springer.

Jie, S.; and Deng, Z.-H. 2023. FacT: Factor-Tuning for lightweight adaptation on Vision Transformer. In *AAAI Conference on Artificial Intelligence*, 1060–1068.

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3D object representations for fine-grained categorization. In *IEEE International Conference on Computer Vision Workshops*, 554–561.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Graf, H. P. 2017. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 1–13.

Lian, D.; Zhou, D.; Feng, J.; and Wang, X. 2022. Scaling & shifting your features: A new baseline for efficient model tuning. In *Advances in Neural Information Processing Systems*, 109–123.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin Transformer: Hierarchical Vision Transformer using shifted windows. In *IEEE/CVF International Conference on Computer Vision*, 9992–10002.

Loshchilov, I.; and Hutter, F. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 1–18.

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

Nilsback, M.-E.; and Zisserman, A. 2008. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*.

Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; and Jawahar, C. V. 2012. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 3498–3505.

Ramachandran, P.; Zoph, B.; and Le, Q. V. 2017. Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Recht, B.; Roelofs, R.; Schmidt, L.; and Shankar, V. 2019. Do ImageNet classifiers generalize to ImageNet? In *International Conference on Machine Learning*, 5389–5400.

Sung, Y.-L.; Cho, J.; and Bansal, M. 2022. LST: Ladder Side-Tuning for parameter and memory efficient transfer learning. In *Advances in Neural Information Processing Systems*, 12991–13005.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the Inception architecture for computer

vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 6000–6010.

Wang, H.; Ge, S.; Lipton, Z.; and Xing, E. P. 2019. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, 10506–10518.

Yun, S.; Han, D.; Chun, S.; Oh, S.; Yoo, Y.; and Choe, J. 2019. CutMix: Regularization strategy to train strong classifiers with localizable features. In *IEEE/CVF International Conference on Computer Vision*, 6022–6031.

Zhai, X.; Puigcerver, J.; Kolesnikov, A.; Ruyssen, P.; Riquelme, C.; Lucic, M.; Djolonga, J.; Pinto, A. S.; Neumann, M.; Dosovitskiy, A.; et al. 2019. A large-scale study of representation learning with the Visual Task Adaptation Benchmark. *arXiv preprint arXiv:1910.04867*.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. *mixup*: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 1–13.

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

Zhou, K.; Liu, Z.; Qiao, Y.; Xiang, T.; and Loy, C. C. 2023. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4): 4396–4415.