GLEAM: Global Share Local Transform MoE for Downstream Transferring With Enhanced Parameter Efficiency

Anonymous submission

Abstract

Parameter-efficient transfer learning (PETL) has emerged as a promising direction to fine-tune lightweight modules and adapt large-scale pre-trained models to downstream tasks. Nevertheless, these methods have not thoroughly explored the characteristics of PETL methods to optimize the fine-tuning performance with miminal volume of parameters. In this paper, we first reveal that, compared to pre-trained models, PETL tends to generate similar features via homogeneous feature transformations across different layers. Subsequently, we propose a Global Sharing Local Transformation framework, namely GLEAM that decomposes the adapter into a shared component and layer-specific local components to simultaneously reduce the redundancy in layer-wise parameter matrices for homogeneous feature transformations and fine-tune the locally specific parameters for minimizing performance loss. Speficially, we develop a shared mixture of convolution that introduces shared multi-scale sparse MoE to enable diverse transformations for suppressing the homogeneity issue of feature transformations in PETL. To accurately evaluate GLEAM, we test it on more than 20 datasets for image classification and few-shot learning performance. Experimental results demonstrate that the proposed method performs on par with existing PETL methods like LoRA with only 3% of its parameters and further yields competitive performance using only 0.07M parameters.

Introduction

Large-scale deep learning models have achieved remarkable success in the fields of natural language processing (Vaswani et al. 2017) and computer vision (Dosovitskiy et al. 2021). However, these models are usually over-parameterized, computationally expensive, and resource-intensive when trained from scratch for each task. To circumvent this issue, the pre-training then fine-tuning paradigm leverages pre-training of large-scale datasets to initialize for specific downstream. Under such paradigm, parameter-efficient transfer learning (PETL) (Hu et al. 2022; Houlsby et al. 2019; Li and Liang 2021; Pfeiffer et al. 2021) further demonstrates significant potential in reducing learnable parameters , consequently garneres extensive attention in the study of large-scale language and vision models.

Existing PETL methods can be categorized into three types, *i.e.*, re-parameterization based, prompt based, and adapter based. Re-parameterization based methods such as LoRA (Hu et al. 2022) decompose model parameters into the product of two trainable low-rank matrices. Prompt based methods like Vision Prompt Tuning (Jia et al. 2022) insert trainable tokens into the input sequence, whereas adapter based methods like Adaptformer (Chen et al. 2022) introduce a parallel small-scale trainable bottleneck into the pre-trained models. Different from the pre-trained large-scale models (Dosovitskiy et al. 2021), we reveal that PETL tends to achieve homogeneous transformations. As demonstrated in Figure 2, we find that the features extracted by PETL module exhibit significantly higher cosine similarity and lower L2 distance. This observation suggests that feature transformations become uniform across different layers during PETL, which may necessitate layer-wise fine-tuning of excessive parameters. Consequently, we resort to propose to weight sharing to enhance parameter efficiency for PETL.

However, as shown in Figure 3, directly applying weight sharing to PETL could exacerbate rather than alleviate homogeneity across different layers, and result in evidently degraded performance on downstream tasks. To address this

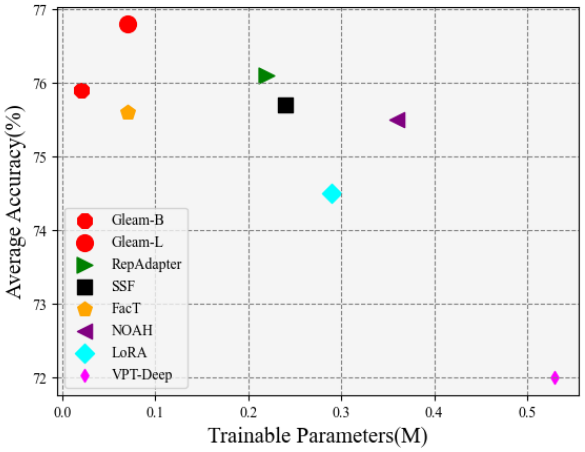

Figure 1: Comparison between GLEAM and other PETL methods on VTAB-1K.For all experiments, we utilize ViT-B pre-trained on ImageNet-21K as base model. GLEAM achieves very competitive performance with a very small number of parameters

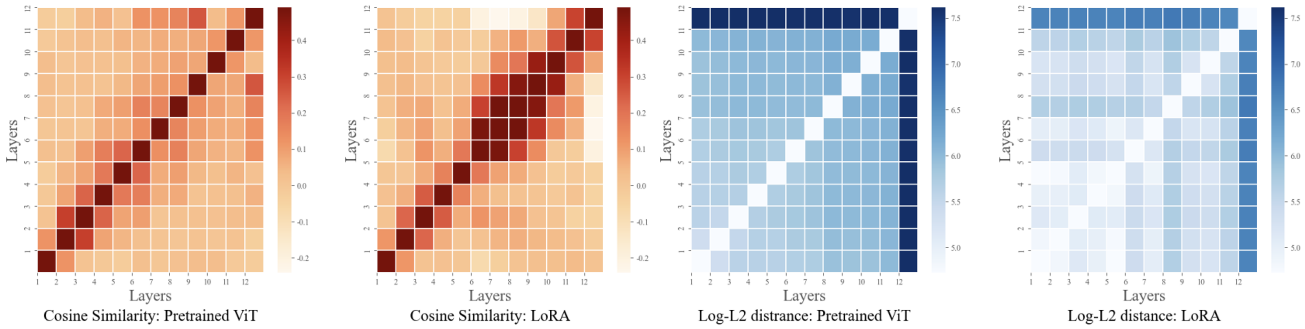

Figure 2: Heatmap of Cosine Similarity and L2 distance between features extracted by ViT and LoRA from different layers. To better accentuate the differences, we set the diagonal elements of the cosine similarity matrices to the common maximum value of the two similarity matrices, while setting the diagonal elements of the L2 distance matrices to the common minimum value of the two distance matrices.It can be observed that compared to the features extracted from the pre-trained model, the features extracted by LoRA exhibit higher cosine similarity as well as lower L2 distance.

problem, we propose a novel PETL framework, namely **GL**obal shar**E** loc**A**l transfor**M** MoE (GLEAM) that globally shares a significant portion of parameters in the adapter modules while applying local transformations to each specific parameter matrix or layer for fine-tuning. We develop a shared form of multi-scale sparse mixture-of-expert (MoE) for PETL to enable diverse transformations and multi-scale feature decoupling with marginally increasing number of parameters. The proposed framework achieves a more favorable trade-off between task performance and storage cost. The contributions of this paper are summarized as below.

- We reveal the issue of homogeneous feature transformations across different layers and demonstrate the feasibility of weight sharing for PETL.
- We propose a **GL**obal shar**E** loc**A**l transfor**M** MoE (GLEAM) framework that globally shares the vast majority of parameters and fine-tunes a small portion of local parameters to adapt to different layers with significantly reduced parameters.
- We develop a shared multi-scale sparse MoE to enable diverse transformations that suppress the homogeneous feature transformations of PETL and achieve better trade-off between storage cost and task performance.

To validate the capabilities of GLEAM, we evaluate it on over 20 datasets (Zhai et al. 2019; Krause et al. 2013; Nilsback and Zisserman 2006; Parkhi et al. 2012; Bossard, Guillaumin, and Van Gool 2014; Maji et al. 2013). Experiments demonstrate that GLEAM could achieve competitive performance while maintain very low storage overhead. Specifically, GLEAM-T using only 0.009M parameters (3% of LoRA) and perform on par with LoRA. Futhermore, GLEAM-B and GLEAM-L with 0.02M and 0.07M parameters could exceed most advanced adapter method.

Related Work

Parameter-Efficient Transfer Learning (PETL) has been widely used in the NLP domain (Houlsby et al. 2019; Hu et al. 2022; Chen et al. 2022; Li and Liang 2021) to adapt models to downstream tasks by fine-tuning a small portion of parameters while keeping the majority of weights fixed. Traditional PETL methods can be categorized into adapter-based, prompt-based, and re-parameterization-based methods. Adapter-based methods (Chen et al. 2022) usually introduce a small module for fine-tuning while keeping the overall model weights fixed. Re-parameterized-based methods such as LoRA (Hu et al. 2022) decompose model parameters into several trainable low-rank matrices and use these low-rank matrices to update the original model parameters. Prompt-based methods (Jia et al. 2022) insert trainable tokens into the input sequence of the model. In this paper, we explore the homogeneity of transformations in PETL modules, which has been overlooked by these methods, decomposing an adapter into shared and local components while introducing sparse multi-scale mixture-of-convolutions to minimize the negative impacts of sharing. As a result, we propose a new high-performance adapter-based PETL method called GLEAM and provides a better trade-off between performance and efficiency.

Mixture of Experts (MoE) (Zhong et al. 2024; Li, Murray, and Carenini 2023; Daxberger et al. 2023; Feng et al. 2023; Shazeer et al. 2017) has been studied for over 30 years, evolving from its initial formulation as an entire model to being integrated as a component within deep neural networks. Despite the variations in its implementation, the core idea remains consistent: training multiple experts specialized in different domains and utilizing a gate module to select the appropriate expert for a given input. The potential of the MoE structure has been gradually uncovered in recent years to handle complex problems with distinctive experts. Particularly, it is demonstrated a significant role in large language models. In this paper, we introduce the concept of MoE into GLEAM to increase the transformation diversity of the PETL module and enhance the representation capability of our model through multi-scale features.

Weight Sharing aims at enhancing parameter efficiency

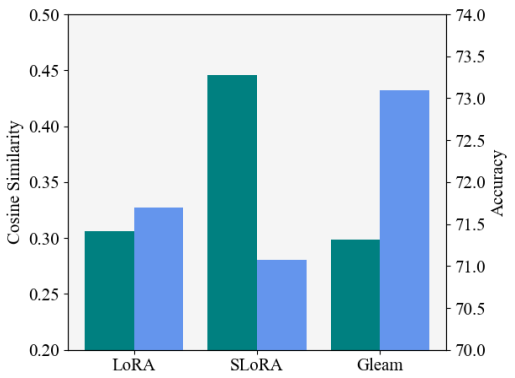

Figure 3: Average cosine similarity between features of adjacent blocks (green columns) and top-1 accuracy (blue columns) on CIFAR100.

by reusing and updating a portion of parameters within a neural network, thereby achieving more efficient utilization of model parameters. Mathematically, weight sharing for a neural network with N repeated layers such as the Transformer can be formulated as $H_{i+1} = f(H_i, \theta)$, where H_{i+1} and H_i denote the input and output of the i-th layer, and θ means the shared parameters across all layers. The core idea of weight sharing has been explored and implemented in NLP and CV domains, including popular BERT, DeiT, and Swin Models (Lan et al. 2020; Zhang et al. 2022; Touvron et al. 2021). In contrast to pre-trained models, weight sharing has not been sufficiently explored for PETL, where weight sharing for parameter reduction without performance loss remains unresolved. In this paper, we demonstrate the feasibility of weight sharing and employ it to further reduce the required parameter count for PETL.

Proposed Method

Preliminaries

In this paper, we focus on the Vision Transformer (ViT) (Dosovitskiy et al. 2021). A basic ViT begins with splitting an input image $I \in \mathbb{R}^{H\times W\times 3}$ into a sequence of fixed-size patches. These patches are then mapped to tokens $x \in \mathbb{R}^{n\times d}$ via a patch embedding layer and concatenated with a learnable classification token $x_{cls} \in \mathbb{R}^{d}$. Positional embeddings $E_{pos} \in \mathbb{R}^{(n+1)\times d}$ are added to preserve positional information. This process can be formulated as:

$$X_0 = [x_{cls}; x_0, x_1, x_2, \ldots, x_l] + E_{pos} \tag{1}$$

X_0 is then fed into ViT blocks that are primarily composed of multi-head self attention (MHSA) (Vaswani et al. 2017) and feed-forward network (FFN). For the l-th ViT block,

$$X'_l = \mathrm{MHSA}(\mathrm{LN}(X_{l-1})) + X_{l-1}, \tag{2}$$

$$X_l = \mathrm{FFN}(\mathrm{LN}(X'_l)) + X'_l, \tag{3}$$

where LN denotes layer normalization. In the MHSA layer, there are four transformations, *i.e.*, query, key, value, and output projection, and N heads are built for each transformation. The parameters for the i-th head include $W_{Q,i}$, $W_{K,i}$, $W_{V,i}$, $W_{O,i} \in \mathbb{R}^{d\times d}$. Similarly, FFN contains two fully-connected layers with parameters $W_{up} \in \mathbb{R}^{d\times 4d}$ and $W_{down} \in \mathbb{R}^{4d\times d}$. For simplicity, we omit the bias terms. MHSA and FFN can be formulated as:

$$\mathrm{MHSA}(X) = \sum_{i=1}^{N} \mathrm{Softmax}\left(\frac{XW_{Q,i}W_{K,i}^T X^T}{\sqrt{d}}\right) XW_{V,i}W_{O,i}, \tag{4}$$

$$\mathrm{FFN}(X) = \mathrm{GELU}(X \cdot W_{up})W_{down}. \tag{5}$$

Motivation: Homogeneous Feature Transformations in PETL

We first reveal that, different from pre-trained models, existing PETL modules achieve homogeneous feature transformations for feature extraction. Most of existing PETL methods map the input features to a low-rank space using a low-dimensional structure to approximate the high-dimensional structure of the pre-trained model. Despite removing redundant information, these methods could discard a large amount of potentially useful information is discarded during fine-tuning and generates features that suffer from insufficient represention capacity for downstream tasks. Subsequently, we provide empirical evidence using the widely used LoRA (Hu et al. 2022).

Given the input x, LoRA can be represented as:

$$h = W_0 x + BAx, \tag{6}$$

where $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$ are the LoRA matrices, $W_0 \in \mathbb{R}^{m\times n}$ denote the pretrained weight, and h is the final output. We apply LoRA to W_O of the MHSA layer in ViT and extract two sets of feature maps, i.e., $W_0 x$ and BAx, from different layers of the fine-tuned model and evaluate their similarity in terms of cosine similarity and L2 distance.

As shown in Figure 2, compared to the feature maps obtained from the pre-trained model, the feature maps transformed by the fine-tuned parameter matrices exhibit higher cosine similarity and lower L2 distance. We take the cosine similarity as an example. The main similarities in the heat map are between features of adjacent blocks. Thus, we further calculate the average cosine similarity between adjacent blocks of LoRA, as shown in Figure 3. Compared with ViT, the average cosine similarity of features extracted by LoRA in adjacent blocks is 0.14 (**84.4%**) higher and the average L2 distance is 210.1 (**59.5%**) lower. This result demonstrates the highly homogeneous feature transformations of PETL compared to the pre-trained model.

This homogeneity is a characteristic of existing PETL approaches, which means the parameter matrices of PETL tend to implement more similar transformations and lead to more similar parameter distributions. Compared to large-scale pre-trained models, the transformations performed by PETL modules are more homogeneous. Meanwhile, to better illustrate the problems brought by weight sharing and how we solve the aggravated homogenization problem caused by weight sharing, we also provide the corresponding similarities and performance of Share LoRA (will be explained in next section) and GLEAM in Figure 3.

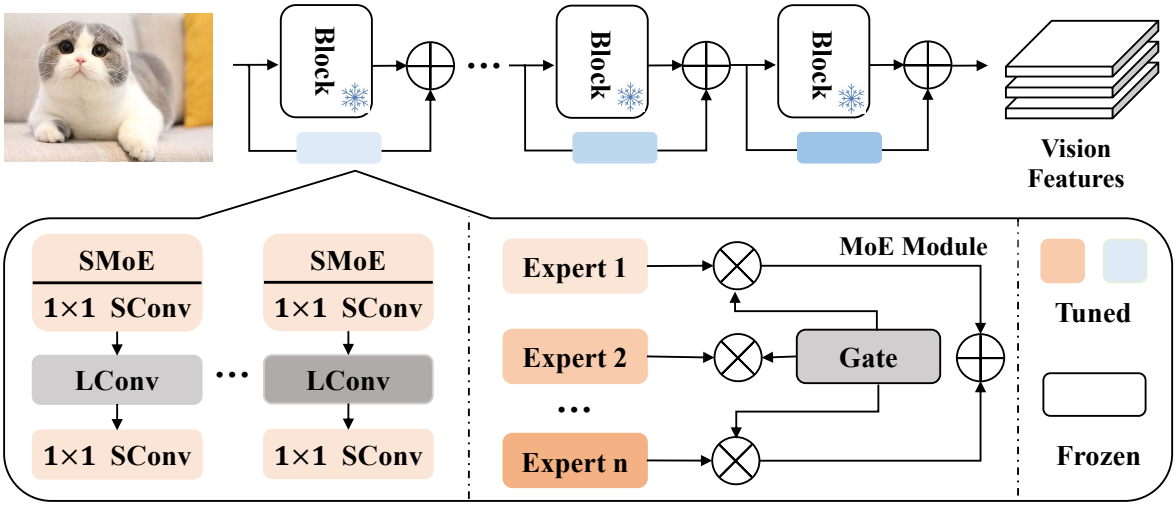

Figure 4: Overall Structure of GLEAM. Modules in white frame except pretrained matrices with the same color means the same parameters. For each pre-trained layer or pre-trained matrix , the convolutional layers for up/down-sampling and the MoE module in GLEAM adopt a global sharing strategy.

GLEAM: Global Share Local Transform MoE

Overview As depicted in Figure 4, GLEAM decomposes the Adapter structure into shared components and introduces an extra local convolution module specifically designed for demultiplexing. This module mitigates the performance loss caused by weight sharing with slightly increasing number of parameters. The shared MoE modules alleviate transformation homogeneity brought by weight sharing and selectively engage the most suitable experts based on the input. See appendix for detailed description of GLEAM.

Feasibility of Weight Sharing in PETL Weight sharing, as a method to improve parameter efficiency, has proven its potential in the NLP domain. Simultaneously, weight sharing is highly aligned with the purpose of PETL to a certain extent. As inferred from the preceding content, PETL is theoretically well-suited for integration with weight sharing. However, applying weight sharing may impact performance to a certain extent. Compared to large-scale and fully trained models, PETL has already undergone a parameter reduction process. It is imperative to investigate the feasibility of weight sharing in PETL and its effect on PETL. Thus, we directly develop a weight-sharing form (SLoRA) for LoRA.

$$h = W_0 x + B' A' x, \tag{7}$$

where B' and A' are global LoRA matrices. However, this direct modification causes a certain degree of performance loss and introduces instability during training. Specifically, when fine-tuning W_O in the MHSA layer for a ViT pretrained on ImageNet-21K and keeping the training settings aligned, using SLoRA led to an accuracy loss of nearly 0.7% on the CIFAR-100 dataset compared to LoRA.

Global Sharing and Local Transformations Although the performance loss of weight sharing on PETL is smaller compared to sharing the pre-trained model, we aim to minimize this performance degradation by introducing a slight storage overhead. We assign a local transformation to each pretrained matrices for demultiplexing(Zhang et al. 2022). Experimental results demonstrate that by incorporating local parameters, the LoRA approach, which simultaneously performs sharing and localization, reduces the original performance loss to 0.3%. Since this local transformation is conducted in a low-dimensional feature space, we argue that introducing a highly limited memory overhead while significantly reducing the performance loss represents a more favorable trade-off between storage and performance. However, although these experiments demonstrate that weight sharing can be applied to PETL at a relatively low cost, as shown in Figure 3, simply sharing weight matrices will inevitably lead to more information loss, thus exacerbating the issue of homogeneity in PETL. Therefore, we need to explore additional methods to mitigate this problem.

Multi-scale Mixture of Experts for Mitigating Homogeneity Based on the aforementioned findings, the issue of exacerbated transformation homogeneity caused by weight sharing necessitates a solution from another perspective. Therefore, to enable PETL to leverage diverse transformations during the fine-tuning process and enhance its understanding of new datasets through features at different scales, we have decided to adopt a shared Mixture of Experts (MoE) module. Numerous studies (Wu et al. 2021; Jie and Deng 2022) have provided compelling evidence supporting the efficacy of integrating convolutional operations into ViTs. Building upon this well-established theoretical foundation, we incorporate visual inductive biases into ViTs while concurrently designing MoE. The entire MoE module consists of a gating module and multiple experts E. To avoid unnecessary parameter overhead, instead of using multiple convolutional kernels of different sizes, each expert in our approach performs interpolation at different ratios on the input feature maps, followed by a convolutional layer of the same

size and average pooling. Given the input z_{in}, the output z_{out} of the MoE module is formulated as:

$$E_i = \text{AvgPool}_i(\text{Conv3}\times3(\text{Upsample}_i(z_{in}))), \quad (8)$$

$$z_{out} = \sum_{i=1}^{n} G_i(z_{in}) \times E_i(z_{in}), \quad (9)$$

where G denotes the gating module with two parameter matrices, *i.e.*, $W_g \in \mathbb{R}^{r\times n}$ and $W_n \in \mathbb{R}^{r\times n}$. W_g is used to learn the gating transformation, while W_n is used to learn the noises. The gating module $G(h)$ is formulated as

$$G(h) = \text{Softmax}(\text{Top-K}(H(h), k)), \quad (10)$$

where

$$H(h) = h \cdot W_g + \mathcal{N}(0,1) \cdot \text{Softplus}(h \cdot W_n), \quad (11)$$

$$\text{Top-K}(v,k) = \begin{cases} v_i, & v_i \text{ is top-}k \text{ elements in } v \\ -\infty, & \text{otherwise} \end{cases}. \quad (12)$$

Experts are selected based on the input features by making the elements beyond the first k elements to infinity and applying the softmax function. Consequently, MoE module chooses the most appropriate feature maps for each input and aggregates them, thereby achieving the previously stated objective of facilitating multi-scale feature extraction.

Experiments

We evaluate different parameter configurations of GLEAM for ViTs and Hierarchical Transformers like Swin-B on the VTAB-1K benchmark (Zhai et al. 2019). VTAB-1K consists of 19 datasets, covering three distinct data domains: natural images (Nature Images), images from remote sensing and medical domains (Special Images), and multiple visual scene understanding datasets (Structure Images). Each dataset contains 1,000 samples divided into a training set of 800 samples and test set of 200 samples. Top-1 accuracy on the test set is reported.

Results for Vision Transformers

Baselines Highly competitive methods are selected as baselines, including BitFit (Zaken, Ravfogel, and Goldberg 2022), Adapter (Houlsby et al. 2019), VPT-shallow (Jia et al. 2022), VPT-deep, Adapter-Former (Chen et al. 2022), LoRA (Hu et al. 2022), NOAH (Zhang, Zhou, and Liu 2024), SSF (Lian et al. 2022), FacT (Jie and Deng 2023), and RepAdapter (Luo et al. 2023). Additionally, we provide a trade-off analysis of our method's performance and storage overhead compared to the existing state-of-the-art method, GLoRA (Chavan et al. 2023). For all the baselines, we set the latent dimension and LoRA rank to 8, while the prompt length for VPT follows the description in (Jia et al. 2022). To showcase the advantages of our method, we also include the performance of full fine-tuning and linear probing.

Implementation Details We use ViT-B/16 pre-trained on ImageNet-21K (Deng et al. 2009) as the base model. The latent dimension is set to 4 for GLEAM-T, 8 for GLEAM-B, and 16 for GLEAM-L. Following Chavan et al. (2023), we use AdamW as the optimizer and employ cosine annealing and warmup scheduling. The batch size is 32 for training. For all tasks, α is selected from $\{0.01, 0.1, 1, 10, 100\}$, the number of experts n is fixed to 4, and the number of selected experts k is searched from $\{1, 2, 3\}$.

Results Figure 1 illustrates the detailed results on VTAB-1K, as summaized below.

i): GLEAM shows highly competitive performance. GLEAM-L using ViT as the backbone only requires 0.07M parameters and outperforms most recent state-of-the-art PETL methods like NOAH, FacT, and RepAdapter on VTAB-1K. Compared to the best RepAdapter, GLEAM-L uses only one-third parameter counts to obtain competitve performance. These results substantiate the effectiveness of our optimization tailored to the characteristics of PETL.

ii): GLEAM-B yields an average accuracy of 75.9% using only 0.02M parameters. It suffers a trivial 0.2% accuracy loss in comparison to RepAdapter and outperforms all the remaining PETL methods. Furthermore, GLEAM-T achieves the same performance as the baseline LoRA using only 3% parameter counts (i.e., 0.009 M vs. 0.29 M). Even with a significant reduction in parameter count, maintaining the existing Global Sharing Local Transformation framework combined with the MoE extension for mitigating homogeneity, we still achieve excellent performance. This result further corroborates that designing PETL methods by considering both transformation similarity and transformation diversity is an effective approach.

In Table 3, we further compare with the state-of-the-art GLoRA (Chavan et al. 2023) that pre-trains an extremely large supernet. We adopt a rank of 1 and keep other experimental settings same to train a supernet with approximately 0.53M parameters and obtain GLoRA with 0.19M parameters. GLoRA suffers from degraded performance and is inferior to GLEAM, when the parameter count further decreases. Note that the GLoRA module is varying in practical applications, since it necessitates pre-training a large supernet for each dataset, a process that incurs considerable memory and time overhead, and requires an extra time-consuming evolutionary search on the supernet. Compared with GLoRA, we flexibly trade-off training time, storage requirements, application feasibility, and performance.

Results for Hierarchical Transformers

We further perform experiments with Swin-B (Liu et al. 2021) as base model. When applying our approach to the Hierarchical Transformer, we maintain global sharing of the MoE modules, while the convolutional layers used for channel modification are adjusted according to the dimensions of each SwinBlock. In other words, this part consists of four shared convolutional layers with different channels. Additionally, we select RepAdapter, which performs best in ViT, as well as the classic VPT, BitFit, full fine-tuning, and linear probing for comparison. Table 4 shows that, despite SwinTransformer incorporates visual biases into consideration and extracts features at different scales through sliding windows, our model still achieves excellent performance on Hierarchical Transformers. Specifically, GLEAM significantly outperforms other baselines and is comparable

Table 1: Full results on VTAB-1K benchmark. GLEAM-B exceeds most existing adavanced PETL method with only 0.02 M trainable parameters while GLEAM-L achieves state-of-the-art performance with 0.07 M trainable parameters.

		Natural							**Specialized**				**Structured**								
Model	Params(M)	Cifar	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 Tuning																					
Full-Tuning	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
Parameter Efficient Transfer Learning																					
BitFit	0.10	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-Shallow	0.06	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	81.7	64.7	49.3	80.3	70.7	45.7	31.7	41.1	74.7
LoRA	0.29	67.1	91.4	69.4	98.8	90.4	83.3	54.0	84.9	93.3	84.4	75.6	**82.9**	**69.2**	49.8	78.5	73.7	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
FacT	0.07	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	0.24	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
RepAdapter	0.22	72.4	91.6	71.0	99.2	91.4	**90.7**	**55.1**	85.3	95.9	84.6	**75.9**	82.3	68.0	50.4	79.9	80.4	49.2	**38.6**	41.0	76.1
Proposed methods																					
GLEAM-T	**0.009**	72.6	90.2	71.2	99.2	90.8	86.4	53.9	84.4	94.8	84.1	75.5	80.2	66.1	46.7	78.2	80.5	47.8	25.9	38.0	74.5
GLEAM-B	0.02	73.0	91.2	71.4	99.2	90.9	88.7	54.3	85.7	95.7	85.7	75.1	81.3	66.7	51.0	80.6	**85.5**	50.6	31.5	40.8	75.9
GLEAM-L	0.07	73.2	92.5	72.2	99.3	91.1	90.5	54.8	85.8	**96.3**	86.4	75.7	82.4	68.1	52.1	81.4	**85.5**	51.9	35.1	42.0	**76.8**

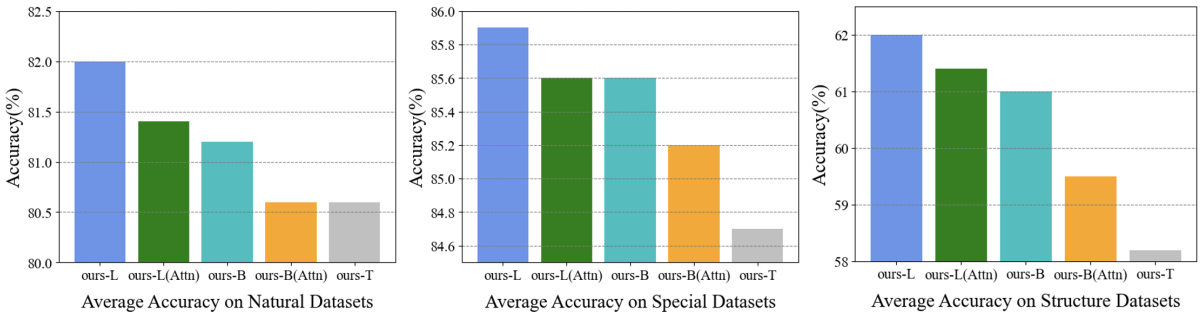

Figure 5: Ablation Study on fine-tuning only attention layers in variants of GLEAM-B and GLEAM-L.

to Repadapter with only a half parameter counts.

Ablation Study

Ablation studies are performed on the VTAB-1K benchmark to validate GLEAM. We explore the impact of weight sharing on attention and linear layers and make sensitivity analysis on hyperparameters n, k, and α.

Impact of Fine-tuning Layer Selection We train variants of GLEAM-L and GLEAM-B that fine-tune only the MHSA layers while keeping the other settings consistent with the previous experiments. As shown in Figure 5, for all experiment groups, full fine-tuning generally outperforms fine-tuning only the attention layers, especially when the hidden dimension is small. Compared to GLEAM-L, the variants fine-tuning only MHSA layers exhibits accuracy loss of 0.8% on Natural Datasets, 0.4% on Special Datasets, and 1.3% on Structure Datasets. For GLEAM-B, the accuracy loss is 0.6%, 0.4%, and 1.5% for the three datasets, respectively. These results suggest that FFN plays a more crucial role in tasks such as visual scene understanding. Another

Table 2: Ablation Study of Expert numbers n, k and scaling factor α. The top row indicates that not use moe module.

DIM	n	k	α	CIFAR100	Caltech101	DTD	Flower102	Oxford-Pets	Avg.
Ablation-Expert number									
16	-	-	1.0	71.5	91.0	69.8	99.3	90.7	84.5
16	2	1	1.0	72.3	91.7	71.1	99.2	90.9	85.0
16	3	1	1.0	72.2	90.7	70.5	99.3	90.8	84.7
16	4	1	1.0	73.1	92.5	72.2	99.3	91.3	85.7
16	5	1	1.0	72.5	90.7	69.1	99.3	90.2	84.4
Ablation-Top-K expert									
16	4	1	1.0	73.1	92.5	72.2	99.3	91.3	85.7
16	4	2	1.0	73.0	91.7	70.4	99.3	91.0	85.1
16	4	3	1.0	73.2	91.7	70.0	99.3	91.1	85.1
Ablation-Scaling factor									
16	4	1	0.01	72.1	90.1	70.2	99.0	90.4	84.4
16	4	1	0.1	72.7	91.0	70.2	99.2	91.0	84.8
16	4	1	1.0	73.1	92.5	72.2	99.3	91.3	85.7
16	4	1	10.0	72.0	90.7	69.6	99.2	90.1	84.3
16	4	1	100.0	70.6	91.3	69.4	99.2	89.8	84.1

Table 3: Performance compared to glora on vtab-1k benchmark. Where Sup. Refers to the number of supernet parameters that GLoRA needs to train, and params refers to the number of parameters after evolutionary search. Nat., Spe.,Str.,Ave. denote Nature Datasets, Special Datasets, Structure Datasets and average accuracy respectively

Model	Sup.(M)	Params(M)	Nat.	Spe.	Str.	Avg.
GLoRA	0.88	0.29	83.4	87.1	61.6	77.3
GLoRA	0.53	0.19	83.1	86.0	59.6	76.2
GLEAM	-	0.07	82.0	86.1	62.3	76.8

Table 4: Results on VTAB-1K with Swin-B as backbone. Our method still exceed full tuning and comparable to RepAdapter with only half learnable parameters.

Model	Params (M)	Natural	Special	Structure	Avg.
Linear	0	73.5	80.8	33.5	62.6
BitFit	0.20	74.2	80.1	42.4	65.6
VPT	0.16	76.8	84.5	53.4	71.6
Full	86.7	79.2	86.2	59.7	75.0
RepAdapter	0.42	82.7	87.5	62.0	77.4
GLEAM	0.21	83.0	86.9	62.2	77.4

noteworthy observation is that when the dimension is larger, fine-tuning only the MHSA layers outperforms fine-tuning both MHSA and FFN with a smaller dimension. This further indicates that, in addition to the choice of pre-trained parameters for fine-tuning, the hidden feature dimension remains one of the key factors influencing performance.

Sensitivity Analysis of Hyperparameters Furthermore, we evaluate the effect of hyperparameters like the number of experts n, k for Top-K operation, and the scaling factor α. Table 2 shows that removing the entire MoE leads to a 1.2% performance loss on the 5 test datasets used. Additionally, too few or too many experts does not significantly improve performance. Especially when n is set to 5 , the MoE module results in a negative gain. k may need to be determined based on the dataset. In most cases 1 yields the best performance, but on datasets such as CIFAR100, performance is better when k is set to 3. Regarding the scaling factor, maintaining α=1.0 could achieves optimal performance on the 5 experimental datasets, while an excessively large or small α leads to a severe performance degradation, which also implies that the performance of GLEAM is relatively sensitive to the selection of hyperparameters.

Conclusion

In this paper, we investigate the similarity of features in PETL and discover that PETL methods tend to implement more similar transformations. Based on this finding, we propose a novel PETL method called GLEAM. The key design of GLEAM lies in simultaneously leveraging the characteristic of similar transformations in PETL by globally sharing a significant portion of the adapter modules, while introducing an MoE module to alleviate the issue of homogeneous transformations in PETL modules. Furthermore, GLEAM achieves excellent performance while maintaining extremely low storage overhead during training and subsequent storage overhead. To validate this, we evaluate the performance of GLEAM on over twenty datasets, and the experimental results demonstrate our better trade-off between storage overhead and performance.

References

Bossard, L.; Guillaumin, M.; and Van Gool, L. 2014. Food-101 – Mining discriminative components with random forests. In *13th European Conference on Computer Vision*, 446–461.

Chavan, A.; Liu, Z.; Gupta, D.; Xing, E.; and Shen, Z. 2023. One-for-All: Generalized LoRA for parameter-efficient fine-tuning. ArXiv:2306.07967 [cs].

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 35*, 16664–16678.

Daxberger, E.; Weers, F.; Zhang, B.; Gunter, T.; Pang, R.; Eichner, M.; Emmersberger, M.; Yang, Y.; Toshev, A.; and Du, X. 2023. Mobile V-MoEs: Scaling down vision transformers via sparse mixture-of-experts. ArXiv:2309.04354 [cs, stat].

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Kai Li; and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

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 *The Ninth International Conference on Learning Representations*.

Feng, Y.; Gong, B.; Jiang, J.; Lv, Y.; Shen, Y.; Zhao, D.; and Zhou, J. 2023. ViM: Vision middleware for unified downstream transferring. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 11662–11673.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; and Morrone, B. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th 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 *The Tenth International Conference on Learning Representations*.

Jia, M.; Tang, L.; Chen, B.-C.; Cardie, C.; Belongie, S.; Hariharan, B.; and Lim, S.-N. 2022. Visual prompt tuning. In *17th European Conference on Computer Vision*, 709–727. Springer Nature Switzerland.

Jie, S.; and Deng, Z.-H. 2022. Convolutional bypasses are better vision transformer adapters. ArXiv:2207.07039 [cs].

Jie, S.; and Deng, Z.-H. 2023. FacT: Factor-tuning for lightweight adaptation on Vision Transformer. In *Proceedings of the 37th 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 *2013 IEEE International Conference on Computer Vision Workshops*, 554–561.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *The Eighth International Conference on Learning Representations*.

Li, R.; Murray, G.; and Carenini, G. 2023. Mixture-of-linguistic-experts adapters for improving and interpreting pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 9456–9469.

Li, X. L.; and Liang, P. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 4582–4597.

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 35*, 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.

Luo, G.; Huang, M.; Zhou, Y.; Sun, X.; Jiang, G.; Wang, Z.; and Ji, R. 2023. Towards efficient visual adaption via structural re-parameterization. ArXiv:2302.08106 [cs].

Maji, S.; Rahtu, E.; Kannala, J.; Blaschko, M.; and Vedaldi, A. 2013. Fine-grained visual classification of aircraft. ArXiv:1306.5151 [cs].

Nilsback, M.-E.; and Zisserman, A. 2006. A visual vocabulary for flower classification. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 1447–1454.

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

Pfeiffer, J.; Kamath, A.; Rücklé, A.; Cho, K.; and Gurevych, I. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, 487–503.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *The Fifth International Conference on Learning Representations*.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, 10347–10357.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, 5998–6008.

Wu, H.; Xiao, B.; Codella, N.; Liu, M.; Dai, X.; Yuan, L.; and Zhang, L. 2021. CvT: Introducing convolutions to Vision Transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 22–31.

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

Zhai, X.; Puigcerver, J.; Kolesnikov, A.; Ruyssen, P.; Riquelme, C.; Lucic, M.; Djolonga, J.; Pinto, A. S.; Neumann, M.; Dosovitskiy, A.; Beyer, L.; Bachem, O.; Tschannen, M.; Michalski, M.; Bousquet, O.; Gelly, S.; and Houlsby, N. 2019. The visual task adaptation benchmark. Https://openreview.net/forum?id=BJena3VtwS.

Zhang, J.; Peng, H.; Wu, K.; Liu, M.; Xiao, B.; Fu, J.; and Yuan, L. 2022. MiniViT: Compressing Vision Transformers with weight multiplexing. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12135–12144.

Zhang, Y.; Zhou, K.; and Liu, Z. 2024. Neural prompt search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Early Access.

Zhong, Z.; Tang, Z.; He, T.; Fang, H.; and Yuan, C. 2024. Convolution meets LoRA: Parameter efficient finetuning for segment anything model. In *The Twelfth International Conference on Learning Representations*.

This paper:

- Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes ✓/partial/no/NA)
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes ✓/no)
- Provides well marked pedagogical references for less-familiare readers to gain background necessary to replicate the paper (yes ✓/no)

Does this paper make theoretical contributions? (yes/no ✓) If yes, please complete the list below.

- All assumptions and restrictions are stated clearly and formally. (yes/partial/no)
- All novel claims are stated formally (e.g., in theorem statements). (yes/partial/no)
- Proofs of all novel claims are included. (yes/partial/no)
- Proof sketches or intuitions are given for complex and/or novel results.(yes/partial/no)
- Appropriate citations to theoretical tools used are given. (yes/partial/no)
- All theoretical claims are demonstrated empirically to hold. (yes/partial/no/NA)
- All experimental code used to eliminate or disprove claims is included. (yes/no/NA)

Does this paper rely on one or more datasets? (yes ✓/no) If yes, please complete the list below.

- A motivation is given for why the experiments are conducted on the selected datasets (yes ✓/partial/no/NA)
- All novel datasets introduced in this paper are included in a data appendix. (yes/partial/no/NA ✓)
- All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes/partial/no/NA ✓)
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. (yes ✓/no/NA)
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. (yes ✓/partial/no/NA)
- All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing. (yes/partial/no/NA ✓)

Does this paper include computational experiments? (yes ✓/no) If yes, please complete the list below.

- Any code required for pre-processing data is included in the appendix. (yes ✓/partial/no).
- All source code required for conducting and analyzing the experiments is included in a code appendix. (yes/partial/no ✓)
- All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes/partial/no ✓)
- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no ✓)
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (yes/partial/no/NA ✓)
- This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. (yes ✓/partial/no)
- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (yes ✓/partial/no)
- This paper states the number of algorithm runs used to compute each reported result. (yes/no ✓)
- Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (yes ✓/no)
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (yes ✓/partial/no)
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (yes/partial/no ✓/NA)
- This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (yes ✓/partial/no/NA)