Revisiting the Parameter Efficiency of Adapters from the Perspective of Precision Redundancy

Shibo Jie Haoqing Wang Zhi-Hong Deng*
School of Intelligence Science and Technology, Peking University
{parsley,wanghaoqing,zhdeng}@pku.edu.cn

Abstract

Current state-of-the-art results in computer vision depend in part on fine-tuning large pre-trained vision models. However, with the exponential growth of model sizes, the conventional full fine-tuning, which needs to store a individual network copy for each tasks, leads to increasingly huge storage and transmission overhead. Adapter-based Parameter-Efficient Tuning (PET) methods address this challenge by tuning lightweight adapters inserted into the frozen pre-trained models. In this paper, we investigate how to make adapters even more efficient, reaching a new minimum size required to store a task-specific fine-tuned network. Inspired by the observation that the parameters of adapters converge at flat local minima, we find that adapters are resistant to noise in parameter space, which means they are also resistant to low numerical precision. To train low-precision adapters, we propose a computational-efficient quantization method which minimizes the quantization error. Through extensive experiments, we find that low-precision adapters exhibit minimal performance degradation, and even 1-bit precision is sufficient for adapters. The experimental results demonstrate that 1-bit adapters outperform all other PET methods on both the VTAB-1K benchmark and few-shot FGVC tasks, while requiring the smallest storage size. Our findings show, for the first time, the significant potential of quantization techniques in PET, providing a general solution to enhance the parameter efficiency of adapter-based PET methods. Code: https://github.com/JieShibo/PETL-ViT

1. Introduction

Large pre-trained vision models have demonstrated exceptional performance on various visual tasks via fine-tuning on task-specific data. In the traditional fine-tuning paradigm, the entire model is updated for each downstream task, resulting in the need to store a fine-tuned model separately for each task. However, with the remarkable scalability of modern vision models, the size of pre-trained vision models is increasing exponentially to achieve superior performance. As a result, the storage cost of the full fine-tuning paradigm becomes prohibitive in multi-task scenarios.

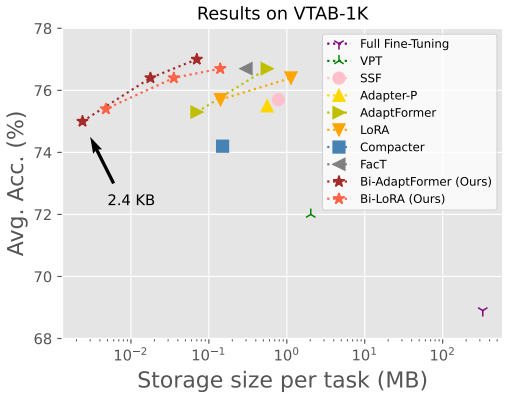

Figure 1. **Average accuracy *vs.* size of trainable parameters in backbones (log scale) on VTAB-1K benchmark.** Our low-precision adapter-based methods outperform other baselines.

Parameter-Efficient Tuning (PET) has recently emerged as a promising approach for fine-tuning a limited number of parameters while attaining performance comparable to full fine-tuning on downstream tasks. Adapter-based methods [5, 18, 19, 23, 24, 38, 43, 45] are among the techniques proposed for PET and have gained considerable attention due to their effectiveness. Adapters are typically small subnetworks with bottleneck architecture comprising two fully-connected (FC) layers inserted into pre-trained models. Adapter-based methods freeze pre-trained weights and update only the adapters, whose parameter efficiency is achieved through their small hidden dimension.

Although the bottleneck adapters have been already lightweight (*e.g.*, 0.5 MB/task for ViT-B [9]), the storage costs remain considerable when dealing with a huge number of tasks (*e.g.*, platform that provides customized models for millions of users). To address this issue, recent stud-

*Corresponding Author.

ies have shown that the parameter efficiency of adapters can be further improved. For example, [17, 24, 38] explore the low-rank structure in adapters, reparameterizing the weight of adapters into smaller subspace with Kronecker, Tensor-Train, or Tucker factorization. Additionally, [16] leverages network pruning to train sparse adapters. We find that these methods actually have a common motivation – reducing the redundancy (*e.g.*, rank redundancy, density redundancy) in adapters. Also motivated by this, we pose a question, *whether there is any other kind of redundancy that can be utilized to better improve the efficiency of adapters.*

In this paper, we begin by exploring the loss landscape of adapters and observe that the local minima of adapters are much flatter than that of the fully fine-tuned models. The flatness of local minima indicates that the trained adapters possess greater resilience to noise in parameter space, such that adapters with low-precision parameters should perform equally well as their high-precision counterparts. Therefore, we infer that adapters are redundant in numerical precision. Since previous work on adapters all employs full-precision (FP32) data type, the impact of precision on adapters has not been investigated yet.

To reduce the precision redundancy, we propose an approach that involves training and storing adapters in low-bit parameter space. Through empirical analysis, we observe that the parameters of each adapter weight approximately follow a Gaussian distribution. Under this assumption, we quantize the adapter parameters by minimizing the quantization loss. Inspired by previous work of neural network quantization [12], we adopt quantization-aware training and train the low-bit adapters with straight-through estimator (STE). Our experiments, conducted on extensive datasets, reveal several key findings: *1)* Unlike quantizing the entire model, quantizing only the adapters results in negligible performance degradation, even in the 1-bit setting; *2)* With a fixed storage budget, 1-bit quantized (*i.e.*, binary) adapters achieve superior performance among all precision settings; *3)* Our 1-bit adapter can outperform all previous PET methods, including low-rank factorization methods, while using the smallest storage size.

Our contributions are summarized as follows:

- From the investigation on the flat local minima of adapters, we infer the existence of precision redundancy in the parameters of adapters, which can be leveraged to improve their parameter efficiency.
- Based on empirical observations of the distribution of adapter parameters, we propose an efficient quantization-aware training method for learning low-bit adapters while minimizing the quantization error.
- Extensive experiments and comparisons verify that lowering the bit-width brings significant efficiency improvement to adapters. Our proposed method achieves new state-of-the-art results in terms of both performance and parameter efficiency.

2. Related Work

Parameter-Efficient Tuning. Parameter-Efficient Tuning (PET) aims to adapt pre-trained vision backbone to downstream tasks by tuning only a small number of parameters. Most work about PET focuses on tuning transformer-based networks, *e.g.*, Vision Transformers (ViTs) [9]. Prompt-based methods [22, 36, 47, 52, 56, 57] concatenate trainable tokens to the sequential inputs of transformers as prompts, adapting the models by tuning the prompts. However, since the computational cost of self-attention is proportional to the square of the length of inputs, prompt-based methods are not as computation-efficient as the original network [5, 22]. Adapter-based methods [5, 14, 18, 19, 23, 24, 37, 38, 43, 45] insert small adapters into the pre-trained model, adjusting the intermediate representations of the network to fit the downstream data. Some of them [19, 24] can be absorbed into the pre-trained weights during inference, which ensures the computational cost is not increased. Besides, there are also methods that tune bias parameters [51], modify the intermediate features via affine transformation [30], fit the change in the network outputs by a small side-network [54], or combine multiple methods automatically [4, 55]. Among them, adapter-based methods have attracted much attention for their competitive performance, generality to different backbones, and scalability.

Efficient Designs of Adapters. As illustrated in Figure 2 (left), adapters are commonly subnetworks composed of two FC layers with nonlinear activation in between. ADAPTER-P [43] places the adapters after the Feed-Forward Network (FFN) blocks, and ADAPTFORMER [5] uses adapters parallel to the FFN blocks of ViT. LORA [19] uses two low-rank matrices to fit the change in the query and value transformation of Multi-Head Self-Attention (MHSA). The formulation of LORA is equivalent to two FC layers without bias parameters and activation, and can be regarded as special adapters in parallel with the query and value weights.

Besides, some work focuses on more compact designs for adapters. COMPACTER [38] and KADAPTATION [17] regard the weights of adapters as the Kronecker product of two smaller matrices, one of which is shared among adapters. FACT [24] tensorizes the network as a tensor, and reparameterizes its change as several factors according to Tensor-Train or Tucker format that are updated end-to-end. Similar to LORA, FACT is not proposed as an adapter-based method, but it can also be viewed as reparameterized adapters with partially shared weights. Besides, SPARSEADAPTER [16] prunes the dense weights of adapters before fine-tuning. These designs reduce the rank

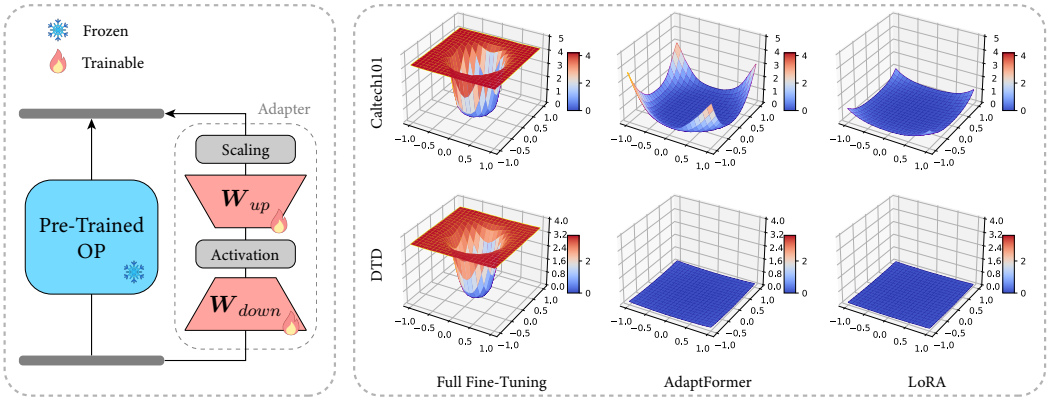

Figure 2. ***Left:*** Illustration of adapters. "Pre-Trained OP" denotes operations in pre-trained models, such as the FFN blocks or QKV transformations in ViTs. ***Right:*** Loss landscape visualization of full fine-tuning and adapter-based tuning [5, 19] on ViT-B.

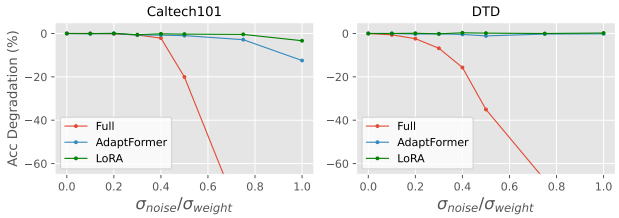

Figure 3. **Accuracy degradation under different intensity of Gaussian noise.** Adapters converge at flatter local minima and are more resistant to disturbation.

and density redundancy in adapters, but we focus on a neglected but more effective direction – precision redundancy.

Network Quantization. Network quantization [12] compresses networks by reducing the bit-width of weight and activation. Current quantization methods include Post-Training Quantization [1, 20, 21, 35, 40, 48, 50], which performs quantization on trained model without re-training; and Quantization-Aware Training [2, 10, 25, 29], which introduce quantization during the training process by approximating the gradient of the non-differentiable quantization operator. The former paradigm does not require access to the entire training data during quantization and has shown almost lossless performance using FP16 and INT8 data type, while the latter yields quantized models with better performance and can work in extremely low-bit settings, *e.g.*, binary quantization [31, 34, 44].

3. Preliminaries

In this paper, we mainly focus on ViTs as pre-trained backbone following previous work [5, 22, 23, 55]. We start with a concise formalization of the commonly used adapters.

ADAPTFORMER [5] uses bottleneck FFN composed of two FC layers with in-between ReLU activation as adapters. The weights of an adapter 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 shortcuts bypassing the FFN blocks, *i.e.*, given an input $\boldsymbol{X} \in \mathbb{R}^{N\times d}$, the computation is formulated as

$$\boldsymbol{X}' = \underbrace{\boldsymbol{X} + FFN(\boldsymbol{X})}_{\text{Frozen}} + \underbrace{s \cdot ReLU(\boldsymbol{X}\boldsymbol{W}_{down})\boldsymbol{W}_{up}}_{\text{Adapter}} \quad (1)$$

where s is a hyper-parameter, $\boldsymbol{X}$ is the input of FFN blocks.

LORA [19] learns the low-rank approximation of change in $\boldsymbol{W}_q$ and $\boldsymbol{W}_v$. Formally, it reparameterizes $\Delta\boldsymbol{W}_{q/v}$ into $\boldsymbol{A}_{q/v}\boldsymbol{B}_{q/v}$, where $\boldsymbol{A}_{q/v} \in \mathbb{R}^{d\times h}$, $\boldsymbol{B}_{q/v} \in \mathbb{R}^{h\times d}$ and $h << d$. The query and value of MHSA are computed as

$$\boldsymbol{Q}/\boldsymbol{V} = \underbrace{\boldsymbol{X}\boldsymbol{W}_{q/v}}_{\text{Frozen}} + \underbrace{s \cdot \boldsymbol{X}\boldsymbol{A}_{q/v}\boldsymbol{B}_{q/v}}_{\text{Adapter}} \quad (2)$$

in which s is a scaling hyper-parameter, and $\boldsymbol{X}$ is the input of MHSA blocks. LORA is equivalent to using ADAPTFORMER-style adapters with identity activation, whose weights are $\boldsymbol{A}_q, \boldsymbol{B}_q, \boldsymbol{A}_v, \boldsymbol{B}_v$.

4. Methodology

4.1. Precision Redundancy in Adapters

It has been extensively studied that the property of a neural network is highly correlated with the flatness of its loss landscape, *e.g.*, the flatter the local minima, the better the generalization [6, 11, 15, 26, 28, 46]. Inspired by them, we here investigate the loss landscape of adapters in vision models to explore their property. Following [28], we plot the loss landscape of full fine-tuning, ADAPTFORMER, and LORA when adapting pre-trained ViT-B [9]. As shown in Figure 2 (right), ADAPTFORMER and LORA obviously converge at much flatter regions than full fine-tuning.

The flat local minima of visual adapters indicate that they generalize better, providing an explanation for their superior

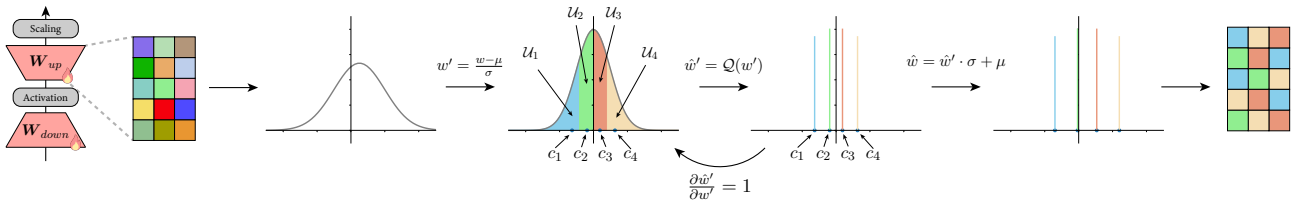

Figure 4. **Illustration of the proposed quantization method with $b = 2$.**

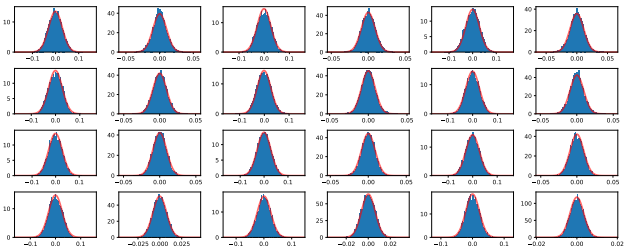

Figure 5. **Parameter frequency histogram visualization** of the 24 weight matrices in all the 12 adapters of ADAPTFORMER fine-tuned on Caltech101. The parameters (blue histograms) are roughly subject to Gaussian distribution (red curves).

performance over full fine-tuning on small and medium-size datasets [22, 23]. Moreover, if the parameters converge at flatter local minima, there are wide low-loss areas around these points. Therefore, when adding noise to the converged parameters, we can expect that the loss will not increase significantly. In other words, the model is resistant to disturbation in parameter space.

As shown in Figure 3, we add Gaussian noise $\mathcal{N}(0, \sigma^2_{noise})$ with different σ_{noise} to the fine-tuned weights, and find that adding noise to adapter-tuned models leads to much less accuracy degradation than fully fine-tuned models. Adapters still retain most of the performance even if the noise has equivalent variance to the weights (*i.e.*, $\sigma_{noise} = \sigma_{weight}$). Since numerical error can be also viewed as a type of noise, we conjecture that the adapters would not suffer from lower numerical precision.

4.2. Trading Precision for Efficiency

In view of the existence of precision redundancy, a natural idea is to trading the redundant precision for much needed efficiency. Previous work on quantization [7, 13, 49] has demonstrated that clustering is a reliable direction for quantization of arbitrary bit-width, so we also adopt a clustering-based quantization strategy for adapters.

As illustrated in Figure 3, the smaller the noise, the less the performance degradation. The object of adapter quantization is to minimize the noise involved, *i.e.*, minimize the quantization error. The b-bit quantization process can be viewed as dividing $\mathbb{R}$ into $B = 2^b$ non-overlapping sets $\{\mathcal{U}_1, ..., \mathcal{U}_B\}$, which correspond to a codebook with B codes $\{c_1, ..., c_B\}$. The quantization function quantizes all values in $\mathcal{U}_j$ to c_j,

$$\mathcal{Q}(w) = c_j \text{ if } w \in \mathcal{U}_j \tag{3}$$

Then we minimize the quantization error as follows,

$$\underset{c_1,...,c_B,\mathcal{U}_1,...,\mathcal{U}_B}{\text{minimize}} \quad \sum_{i=1}^{m} |w_i - \mathcal{Q}(w_i)|^p \tag{4}$$

in which w_i is an element of a weight $\boldsymbol{W}$ of the adapters, and m is the number of elements in $\boldsymbol{W}$. This problem is equivalent to 1D clustering, which can be addressed via clustering algorithm such as k-means ($p = 2$) and k-medians ($p = 1$).

Low-bit quantization, particularly 1-bit quantization suffers catastrophically poor performance in the absence of quantization-aware training (QAT). In QAT, the weights are ever-changing, so the clustering algorithm has to be rerun in each forward propagation during tuning. An appropriate clustering algorithm is supposed to have negligible computational cost, but an iterative algorithm like k-means and k-medians is not efficient enough. Moreover, since the cluster assignment in k-means and k-medians is not differentiable, this process cannot be end-to-end optimized in QAT. Therefore, although previous work [13] has applied k-means into post-training quantization, it is not a suitable choice for QAT on adapters.

To find an efficient and differentiable clustering method, we visualize the frequency histogram of the parameters in the weights of adapters. As shown in Figure 5, we find that the parameters in full-precision adapters are subject to a bell-shaped distribution with tails. For simplicity, we suppose the parameters of each weight are always Gaussian, so that the clustering algorithm can be simplified considerably.

Before tuning, we perform clustering on a standard Gaussian distribution to calculate $\{c_1, ..., c_B\}$ and $\{\mathcal{U}_1, ..., \mathcal{U}_B\}$. We suppose $p = 1$ in Eq. (4) and use k-medians for simplicity. As illustrated in Figure 4, in each training step, we first standardize the weights by the means and variances of their parameters,

$$w'_i = \frac{w_i - \mu}{\sigma} \tag{5}$$

where $\mu = MEAN(\{w_i\}_{i=1}^m), \sigma = STD(\{w_i\}_{i=1}^m)$. According to the Gaussian assumption, the parameters in each stan-

dardized weight are subject to standard Gaussian distribution. Then we quantize each standardized weight with the pre-calculated $\{c_1, ..., c_B\}$ and $\{\mathcal{U}_1, ..., \mathcal{U}_B\}$,

$$\hat{w_i}' = \mathcal{Q}(w_i') = c_j \text{ if } w_i' \in \mathcal{U}_j \tag{6}$$

Finally, we de-standardize the weights to their original means and variances,

$$\hat{w_i} = \hat{w_i}' \cdot \sigma + \mu \tag{7}$$

and then feed the inputs to perform the forward and backward propagation.

In the whole quantization process, only the quantization operation $\mathcal{Q}$ is not differentiable, so we use straight-through estimator (STE) to approximate the gradient, *i.e.*, $\frac{\partial \mathcal{Q}(w_i')}{\partial w_i'} = 1$. Then $\forall w_i, w_k \in \boldsymbol{W}$ the overall gradient is calculated as

$$\frac{\partial \hat{w_i}}{\partial w_k} = \begin{cases} 1 + \frac{w_i'(\hat{w_i}' - w_i')}{m} & \text{if } i = k \\ \frac{w_k'(\hat{w_i}' - w_i')}{m} & \text{otherwise} \end{cases} \tag{8}$$

During tuning, the pre-trained weights are always frozen, and only the adapters as well as the classification head are updated. The full-precision weights are maintained in training, and updated via end-to-end gradient descent. Since PET only focuses on boosting parameter efficiency, we still use full-precision activation for better performance. After tuning, we store necessary information for reproducing the quantized weights instead of the full-precision adapters, *i.e.*, the b-bit quantization indexes j of adapters' parameters (b bits per parameter) and the mean μ and standard deviation σ of each full-precision weight matrix in adapters (128 bits per adapter). $\{c_1, ..., c_B\}$ and $\{\mathcal{U}_1, ..., \mathcal{U}_B\}$ can be recalculated before inference. At inference time, the weights are reconstructed as

$$\hat{w_i} = c_j \cdot \sigma + \mu \tag{9}$$

which are directly used for inference.

5. Experiments

5.1. Datasets

We use more than 20 image classification tasks to evaluate the performance of different PET methods.

VTAB-1K benchmark. VTAB-1K [53] contains 19 image classification tasks from diverse fields, which can be categorized into three groups: Natural, Specialized, and Structured. These tasks cover a large range of possible domains where downstream tasks come, so the performance of different methods on this benchmark largely reflects their ability to transfer learning. Each dataset contains 800 samples for training and 200 for validation. Following previous work [22–24, 30, 55], we tune the pre-trained model with all the 1,000 training and validation samples and report results evaluated on test-set. Following [22, 30], we use *unnormalized inputs* that are consistent with the VTAB paper [53]. Note that some previous methods [24, 55] normalize the images with ImageNet's mean and standard deviation, so we re-implement some of them for a fair comparison.

Few-shot fine-grained visual recognition (FGVC). We use five FGVC datasets to evaluate the capability of PET methods in the low-data regime. The five datasets are FGVC-Aircraft [39], Oxford-Pets [42], Food-101 [3], Stanford Cars [27], and Oxford-Flowers102 [41]. Experiments are conducted in 1, 2, 4, 8, and 16-shot settings.

Method	Bit-width	Avg. Acc.	Size (KB)
ADAPTFORMER	*32 (FP)*	76.70	576.0
	16 (FP)	76.70 (- 0.00)	288.0
	8 (INT)	76.69 (↓ 0.01)	144.0
	4	76.76 (↑ 0.06)	72.3
	2	76.64 (↓ 0.06)	36.2
	1	76.41 (↓ 0.29)	18.2
LORA	*32 (FP)*	76.42	1152.0
	16 (FP)	76.42 (- 0.00)	576.0
	8 (INT)	76.42 (- 0.00)	288.0
	4	76.33 (↓ 0.09)	144.4
	2	76.27 (↓ 0.15)	72.4
	1	76.40 (↓ 0.02)	36.4

Table 1. **Average accuracy on VTAB-1K benchmark.** We fix $h = 8$ for ADAPTFORMER and LORA and change the bit-width. "Size" denotes the size of adapters per task.

Method	Bit-width	Dimension	Avg. Acc.
ADAPTFORMER	*32 (FP)*	1	75.29
	8 (INT)	4	76.34 (↑ 1.05)
	4	8	76.76 (↑ 1.47)
	2	16	76.89 (↑ 1.60)
	1	32	**76.97** (↑ 1.68)
LORA	*32 (FP)*	1	75.70
	8 (INT)	4	76.08 (↑ 0.38)
	4	8	76.33 (↑ 0.63)
	2	16	76.70 (↑ 1.00)
	1	32	**76.72** (↑ 1.02)

Table 2. **Average accuracy on VTAB-1K benchmark under certain storage budget.** Lower bit-width and higher hidden dimension lead to better performance.

5.2. Performance of Low-Precision Adapters

We first address the most critical question in this paper: *is reducing precision redundancy a good choice for improving the parameter efficiency of adapter-based PET methods?* To investigate the role of numerical precision in adapters, we make comparisons across different bit-widths. We use ADAPTFORMER [5] and LORA [19] with $h = 8$ to adapt ViT-B/16 [9] pre-trained on supervised ImageNet-21K [8]. The 32-bit adapters are trained using FP32 without quantization. 16-bit (FP16) and 8-bit (INT8) adapters

	Size (MB)	Avg. Acc.	Natural							Specialized				Structured							
			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
Conventional Fine-Tuning																					
FULL	327	68.9	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
LINEAR	0	57.6	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
PET methods																					
VPT-DEEP [22]	2.03	72.0	**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
NOAH† [55]	1.37	75.5	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
LORA [19]	1.13	76.4	72.0	91.2	71.6	99.1	91.3	88.9	56.4	87.2	94.6	83.9	74.9	**83.7**	64.0	52.3	81.2	84.8	53.3	**38.1**	43.4
SSF [30]	0.78	75.7	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
ADAPTER-P [43]	0.56	75.5	73.2	90.1	69.6	99.2	91.1	84.9	56.0	86.6	94.8	82.5	**75.8**	82.9	63.9	49.7	79.7	81.7	55.5	31.6	42.2
ADAPTFORMER [5]	0.56	**76.7**	73.8	92.3	72.7	99.3	91.6	89.1	56.5	**87.8**	95.5	84.9	75.2	83.3	62.5	52.4	**81.7**	86.2	**55.9**	34.4	40.2
BITFIT [51]	0.39	65.2	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
FACT-TT [24]	0.30	**76.7**	73.4	91.0	72.4	99.2	91.4	90.1	**56.6**	87.3	94.7	84.5	**75.8**	83.0	64.9	51.3	81.4	**87.4**	53.2	33.5	**44.3**
VPT-SHALLOW [22]	0.24	67.8	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
COMPACTER [38]	**0.15**	74.2	71.9	89.0	69.7	99.1	90.7	82.7	56.1	86.0	93.5	82.4	75.3	80.2	63.4	47.4	77.2	78.1	53.5	27.3	39.8
BI-LORA (Ours)																					
$h = 32$	0.14	76.7	72.1	91.7	71.2	99.1	91.4	**90.2**	55.8	87.0	**95.4**	85.5	75.5	83.1	64.1	52.2	81.3	**86.4**	53.5	**36.7**	**44.4**
$h = 1$	0.0048	75.4	72.6	90.4	71.8	99.0	91.3	87.0	56.0	86.1	94.1	82.1	75.4	81.0	**64.2**	50.5	79.7	83.0	53.7	29.7	42.9
BI-ADAPTFORMER (Ours)																					
$h = 32$	0.071	**77.0**	**74.1**	**92.4**	**72.1**	**99.3**	**91.6**	89.0	**56.3**	**88.2**	95.2	**86.0**	**76.2**	**83.9**	63.6	**53.0**	**81.4**	86.2	**54.8**	35.2	41.3
$h = 1$	**0.0024**	75.0	73.3	91.0	**72.1**	99.1	91.4	86.0	56.2	87.0	94.6	82.9	76.0	79.6	62.8	50.1	78.6	76.6	53.9	27.4	38.6

Table 3. **Full results on the VTAB-1K benchmark**. "Avg. Acc." denotes the average results over three groups. "Size" denotes the average size of trainable parameters in backbones per task, *i.e.*, classification heads (0.14 MB/task in average) are not counted. † denotes results from [55] using normalized inputs.

are directly converted from fine-tuned FP32 adapters. Others are fine-tuned using the proposed QAT method. Table 1 presents the accuracy and adapter size on VTAB-1K.

We notice that using b-bit adapters leads to about $\frac{32}{b}\times$ more parameter efficiency than full-precision adapters. However, the performance degradation resulting from quantization is very slight and sometimes negligible, even in the 1-bit setting. Note that quantizing the entire model to a very low bit-width usually causes significant performance degradation, but our observation indicates that low-bit quantization only on adapters is reliable and much less damaging.

Moreover, we explore the best bit-width given a certain storage budget. Since low-precision adapters are more lightweight, we can augment their performance by using higher hidden dimension to utilize the saved space. The size of a b-bit h-dimension adapter is about $2dbh$ bits where d is the feature dimension, so we fix $bh = 32$ and compare different combinations of b and h. As shown in Table 2, the lower b and higher h yield better performance on LORA and ADAPTFORMER. 1-bit adapters perform the best across different combinations. Overall, we find that the parameter efficiency gains of the low-bit adapters far outweigh their performance damage, demonstrating the feasibility and necessity to trade precision for efficiency.

5.3. Comparison with the State-of-the-Art

5.3.1 VTAB-1K benchmark

We compare our methods with full fine-tuning, linear probing (*i.e.*, only training the classification head), VPT [22], NOAH [55], SSF [30], ADAPTER-P [43], BITFIT [51], ADAPTFORMER [5], LORA [19], COMPACTER [38], and FACT [24] on VTAB-1K. All baselines use FP32 by default. The hidden dimension h is set to 8 for ADAPTER-P, ADAPTFORMER, and LORA. The number of Kronecker products and hidden dimensions are 4 and 32 for COMPACTER, respectively. For FACT, we use FACT-TT with rank searched from $\{8, 16, 32\}$ to adapt the MHSA blocks. The settings of other baselines follow their original papers. As for our low-precision adapters, we quantize the bit-width of ADAPTFORMER and LORA to 1, named BI-ADAPTFORMER and BI-LORA, and report results with hidden dimensions $h = 1$ and 32. All these methods use a ViT-B/16 [9] pre-trained on supervised ImangeNet-21K as backbone. We train the models for 100 epochs with AdamW optimizer.

Table 3 shows the full results on VTAB-1K. Since 1-bit adapters are much more storage-efficient than their full-precision counterparts, BI-ADAPTFORMER and BI-LORA can use a larger hidden dimension while maintaining a

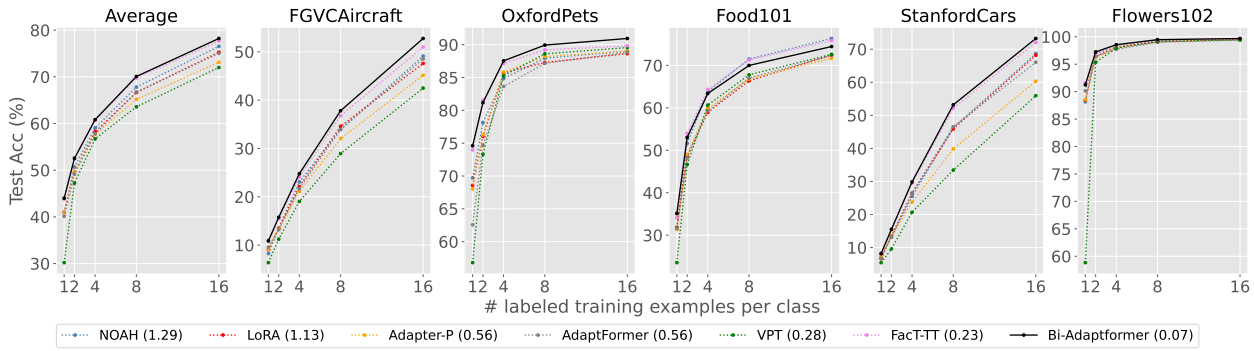

Figure 6. **Accuracy of few-shot learning on FGVC datasets.** The average size (MB) of trainable parameters in backbones is shown in parentheses. BI-ADAPTFORMER outperforms other baselines on average accuracy using the fewest trainable parameters. Results are averaged over three trials with different seeds.

Method	h	Binary Head	Avg. Acc.	Ckpt Size (KB)
FULL	-	×	68.9	3.4×10^5
LINEAR	-	×	57.6	140.8
BI-ADAPTFORMER	32	×	76.97	212.9
	32	✓	76.89 (↓ 0.08)	76.6
	1	×	74.96	143.2
	1	✓	73.81 (↓ 1.15)	**6.8**
BI-LORA	32	×	76.72	285.1
	32	✓	76.31 (↓ 0.41)	148.8
	1	×	75.39	145.6
	1	✓	74.56 (↓ 0.83)	**9.2**

(a) **Classification head quantization.** "Ckpt Size" denotes the average size of checkpoint including classification heads.

Backbone	Method	Avg. Acc.	Size (MB)
ConvNeXt-B	FULL	74.03	334.0
	LINEAR	63.58	0
	VPT-DEEP	68.71	0.017
	ADAPTFORMER	78.86	1.102
	BI-ADAPTFORMER	**79.07** (↑ 0.21)	0.138
Swin-B	Full	74.99	332.2
	Linear	62.60	0
	VPT-DEEP	71.55	0.622
	ADAPTFORMER	77.22	0.734
	BI-ADAPTFORMER	**77.24** (↑ 0.02)	0.092

(b) **Performance on other backbones.** We use ConvNeXt-B and Swin-B as backbones.

Method	Training Time (ms/batch)	Inference Time (ms/batch)
FULL	342.0	**110.8**
LINEAR	**115.2**	**110.8**
VPT-DEEP	407.9	174.2
FACT-TT	296.0	**110.8**
ADAPTFORMER	252.3	115.3
BI-ADAPTFORMER	265.0 (↑ 12.7)	115.5 (↑ 0.02)
LORA	275.7	**110.8**
BI-LORA	293.2 (↑ 17.5)	**110.8**

(c) **Average training and inference time.** Measured on a single GeForce RTX 3090 GPU with batch size 64.

Method	Bit-width	Avg. Acc.
ADAPTFORMER	2 (PTQ)	74.89
	1 (PTQ)	67.28
	2 (Ours)	**76.64** (↑ 1.75)
	1 (Ours)	**76.41** (↑ 9.13)
LORA	2 (PTQ)	74.22
	1 (PTQ)	67.38
	2 (Ours)	**76.27** (↑ 2.05)
	1 (Ours)	**76.40** (↑ 9.02)

(d) **QAT *vs.* PTQ.** "PTQ" denotes directly quantizing fine-tuned FP32 adapters using k-means.

Method	# Block	Avg. Acc.	Size (KB)
BI-ADAPTFORMER	1	**76.97**	**72.2**
	8	76.96	73.5
	32	76.92	78.0
BI-LORA	1	**76.72**	**144.4**
	8	76.69	147.0
	32	76.66	156.0

(e) **Block-wise quantization.** We use BI-ADAPTFORMER and BI-LORA with $h = 32$.

Table 4. **Supplementary results on VTAB-1K benchmark.**

smaller size. Our BI-ADAPTFORMER with $h = 32$ beats all previous PET methods while using a smaller storage size. Notably, BI-ADAPTFORMER and BI-LORA achieve better performance than COMPACTER and FACT-TT while being more parameter-efficient, indicating that precision redundancy is more significant than rank redundancy in adapters and thus quantization is a better solution than low-rank parameterization for designing efficient adapters. Moreover, BI-ADAPTFORMER and BI-LORA with $h = 1$ only store less than 5 KB of backbone parameters for each task, while reaching performance better than VPT, BITFIT, COMPACTER, and full fine-tuning.

5.3.2 Few-shot learning on FGVC

On few-shot FGVC datasets, we compare BI-ADAPTFORMER, the best-performing quantized adapter in the experiments above, with other competitive baselines: VPT-DEEP, ADAPTER-P, LORA, ADAPTFORMER, NOAH, and FACT-TT. The hidden dimensions of ADAPTER-P, LORA, and ADAPTFORMER, as well as the

prompt length of VPT-DEEP, are all set to 8. The rank of FACT-TT is set to 16, and NOAH follows the best recipes in [55]. As for BI-ADAPTFORMER, we use a hidden dimension of 32. Other settings are the same as in the VTAB-1K experiments. Per-dataset results as well as the average results in the five settings are shown in Figure 6.

Overall, our BI-ADAPTFORMER outperforms all baselines on 5-task average accuracy with the smallest size of trainable parameters. On FGVC-Aircraft, Oxford-Pets, and Stanford Cars, BI-ADAPTFORMER exhibits significant performance improvement over the previously state-of-the-art PET methods. Only on Food-101, BI-ADAPTFORMER performs worse than FACT-TT and NOAH. Note that BI-ADAPTFORMER is about 3× and 19× more storage-efficient than FACT and NOAH, respectively, and thus is more competitive under strict storage restrictions.

5.4. Further Analysis

5.4.1 Quantizing classification head

As the size of adapters is compressed, the classification heads take up most of the storage space, hindering further improvements in storage efficiency. For example, on VTAB-1K, the average size of the classification heads is 0.14 MB, much larger than that of BI-ADAPTFORMER modules. As shown in Table 4a, by quantizing the classification heads, BI-ADAPTFORMER keeps state-of-the-art results (76.89 *vs.* ADAPTFORMER's 76.70) with checkpoint size smaller than linear probing (76.7 KB *vs.* 140.8 KB). Note that linear probing is usually considered as the efficiency lower bound of adaptation. Furthermore, BI-ADAPTFORMER and BI-LORA with $h = 1$ and binary head achieve better performance than full fine-tuning, linear probing, and VPT, but the average size of the total checkpoints is only 6.8 KB and 9.2 KB, respectively, which are dozens of times more storage-efficient than linear probing.

5.4.2 Computational efficiency

One of the design principles behind our quantization method is to ensure the quantization operation has negligible computational cost during QAT. To evaluate the efficiency of our proposed method, we conduct experiments to study the training and inference time of different tuning methods, as summarized in Table 4c. For all baselines, we use the same settings as in the VTAB-1K experiments. As for our BI-ADAPTFORMER and BI-LORA, we set a larger hidden dimension $h = 32$. We find that the QAT and larger h slightly increase the training time of adapters. However, BI-ADAPTFORMER and BI-LORA are still faster than VPT, FACT, and full fine-tuning. At inference time, since (BI-)LORA, and FACT can be re-parameterized and absorbed into the pre-trained backbone, they do not incur additional computation.

5.4.3 Performance on other backbones

Note that our proposed quantization method is a plug-in strategy that can be applied in any backbones and any adapters. Besides ViTs [9], there are also other commonly used backbone networks in vision, such as hierarchical transformers like Swin [32] and convolutional networks like ConvNeXt [33]. In Table 4b, we apply BI-ADAPTFORMER to Swin-B and ConvNeXt-B, and compare it with other baselines that can also be extended to these backbones. We notice that BI-ADAPTFORMER still achieves state-of-the-art results on VTAB-1K. BI-ADAPTFORMER with $h = 32$ offers on-par or better performance than ADAPTFORMER with $h = 8$ while only using about $\frac{1}{8}$ of the storage size, which verifies the generalization ability of binary adapters.

5.4.4 Ablation studies

We perform further ablation experiments on our low-bit adapters. The low-bit adapters are fine-tuned via QAT, which has been proven to work better in low-bit settings. To illustrate this, we compare our method with a PTQ method, *i.e.*, directly quantizing fine-tuned full-precision adapters using k-means. We set $h = 8$ for ADAPTFORMER and LORA. As shown in Table 4d, PTQ obviously underperforms QAT, especially in 1-bit setting.

Moreover, since each weight matrix can be divided into several sub-matrices as blocks to perform block-wise quantization, *i.e.*, standardizing the parameters and storing the μ and σ of each block, we here compare the performance of 1-bit adapters across different numbers of blocks. We set $h = 32$ for all methods. As shown in Table 4e, since block-wise quantization methods (# block > 1) store more μ and σ than our methods (# block $= 1$), block-wise quantization uses a larger storage size. However, block-wise quantization does not demonstrate superiority over our methods.

6. Conclusion

In this work, we systematically revisit the parameter efficiency of adapter-based PET through the lens of precision redundancy. Based on our observations, we propose a plug-in strategy to train low-precision counterparts for existing adapter-based methods. Through extensive experiments on more than 20 datasets, we empirically verify the superiority of 1-bit adapters in terms of both performance and parameter efficiency. Surprisingly, we find that 2.4 KB parameters in backbone is almost sufficient to describe the difference between the pre-trained ViT-B and a task-specific fine-tuned ViT-B, suggesting that the intrinsic dimension of visual datasets is much smaller than what we used to believe. Our work also brings quantization to PET, providing a general solution to largely enhance the parameter efficiency of adapter-based PET methods.

References

[1] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Proceedings of NeurIPS*, 2019. 3

[2] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. LSQ+: improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of CVPR*, 2020. 3

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Proceedings of ECCV*, 2014. 5

[4] Arnav Chavan, Zhuang Liu, Deepak K. Gupta, Eric P. Xing, and Zhiqiang Shen. One-for-all: Generalized lora for parameter-efficient fine-tuning. *arXiv preprint*, arXiv:2306.07967, 2023. 2

[5] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. In *Proceedings of NeurIPS*, 2022. 1, 2, 3, 5, 6

[6] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pre-training or strong data augmentations. In *Proceedings of ICLR*, 2022. 3

[7] Minsik Cho, Keivan Alizadeh-Vahid, Saurabh Adya, and Mohammad Rastegari. DKM: differentiable k-means clustering layer for neural network compression. In *Proceedings of ICLR*, 2022. 4

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of CVPR*, 2009. 5

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of ICLR*, 2021. 1, 2, 3, 5, 6, 8

[10] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *Proceedings of ICLR*, 2020. 3

[11] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *Proceedings of ICLR*, 2021. 3

[12] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint*, arXiv:2103.13630, 2021. 2, 3

[13] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *Proceedings of ICLR*, 2016. 4

[14] Tianxiang Hao, Hui Chen, Yuchen Guo, and Guiguang Ding. Consolidator: Mergable adapter with group connections for visual adaptation. In *Proceedings of ICLR*, 2023. 2

[15] Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. On the effectiveness of adapter-based tuning for pretrained language model adaptation. In *Proceedings of ACL/IJCNLP*, 2021. 3

[16] Shwai He, Liang Ding, Daize Dong, Jeremy Zhang, and Dacheng Tao. Sparseadapter: An easy approach for improving the parameter-efficiency of adapters. In *Findings of EMNLP*, 2022. 2

[17] Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. Parameter-efficient model adaptation for vision transformers. In *Proceedings of AAAI*, 2023. 2

[18] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of ICML*, 2019. 1, 2

[19] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *Proceedings of ICLR*, 2022. 1, 2, 3, 5, 6

[20] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *Proceedings of ICML*, 2021. 3

[21] Yongkweon Jeon, Chungman Lee, Eulrang Cho, and Yeonju Ro. Mr.biq: Post-training non-uniform quantization based on minimizing the reconstruction error. In *Proceedings of CVPR*, 2022. 3

[22] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge J. Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *Proceedings of ECCV*, 2022. 2, 3, 4, 5, 6

[23] Shibo Jie and Zhi-Hong Deng. Convolutional bypasses are better vision transformer adapters. *arXiv preprint*, arXiv:2207.07039, 2022. 1, 2, 3, 4, 5

[24] Shibo Jie and Zhi-Hong Deng. Fact: Factor-tuning for lightweight adaptation on vision transformer. In *Proceedings of AAAI*, 2023. 1, 2, 5, 6

[25] Sangil Jung, Changyong Son, Seohyung Lee, JinWoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of CVPR*, 2019. 3

[26] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *Proceedings of ICLR*, 2017. 3

[27] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of CVPR workshops*, 2013. 5

[28] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Proceedings of NeurIPS*, 2018. 3

[29] Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. Q-vit: Accurate and fully quantized low-bit vision transformer. In *Proceedings of NeurIPS*, 2022. 3

[30] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. In *Proceedings of NeurIPS*, 2022. 2, 5, 6

[31] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Proceedings of NIPS*, 2017. 3

[32] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of ICCV*, 2021. 8

[33] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of CVPR*, 2022. 8

[34] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *Proceedings of ECCV*, 2020. 3

[35] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. In *Proceedings of NeurIPS*, 2021. 3

[36] Yuning Lu, Jianzhuang Liu, Yonggang Zhang, Yajing Liu, and Xinmei Tian. Prompt distribution learning. In *Proceedings of CVPR*, 2022. 2

[37] Gen Luo, Minglang Huang, Yiyi Zhou, Xiaoshuai Sun, Guannan Jiang, Zhiyu Wang, and Rongrong Ji. Towards efficient visual adaption via structural re-parameterization. *arXiv preprint*, arXiv:2302.08106, 2023. 2

[38] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *Proceedings of NeurIPS*, 2021. 1, 2, 6

[39] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint*, arXiv:1306.5151, 2013. 5

[40] Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *Proceedings of ICML*, 2020. 3

[41] M-E Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *Proceedings of CVPR*, 2006. 5

[42] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *Proceedings of CVPR*, 2012. 5

[43] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of EACL*, 2021. 1, 2, 6

[44] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognit.*, 2020. 3

[45] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Proceedings of NIPS*, 2017. 1, 2

[46] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Proceedings of NeurIPS*, 2018. 3

[47] Sheng Shen, Shijia Yang, Tianjun Zhang, Bohan Zhai, Joseph E. Gonzalez, Kurt Keutzer, and Trevor Darrell. Multitask vision-language prompt tuning. *arXiv preprint*, arXiv:2211.11720, 2022. 2

[48] Peisong Wang, Qiang Chen, Xiangyu He, and Jian Cheng. Towards accurate post-training network quantization via bit-split and stitching. In *Proceedings of ICML*, 2020. 3

[49] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-Sheng Hua. Quantization networks. In *Proceedings of CVPR*, 2019. 4

[50] Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization. In *Proceedings of ECCV*, 2022. 3

[51] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of ACL*, 2022. 2, 6

[52] Yuhang Zang, Wei Li, Kaiyang Zhou, Chen Huang, and Chen Change Loy. Unified vision and language prompt learning. *arXiv preprint*, arXiv:2210.07225, 2022. 2

[53] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, André Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. The visual task adaptation benchmark. *arXiv preprint*, arXiv:1910.04867, 2019. 5

[54] Jeffrey O. Zhang, Alexander Sax, Amir Roshan Zamir, Leonidas J. Guibas, and Jitendra Malik. Side-tuning: A baseline for network adaptation via additive side networks. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Proceedings of ECCV*, 2020. 2

[55] Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. Neural prompt search. *arXiv preprint*, arXiv:2206.04673, 2022. 2, 3, 5, 6, 8

[56] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *arXiv preprint*, arXiv:2109.01134, 2021. 2

[57] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of CVPR*, 2022. 2