Sensitivity-Aware Efficient Fine-Tuning via Compact Dynamic-Rank Adaptation

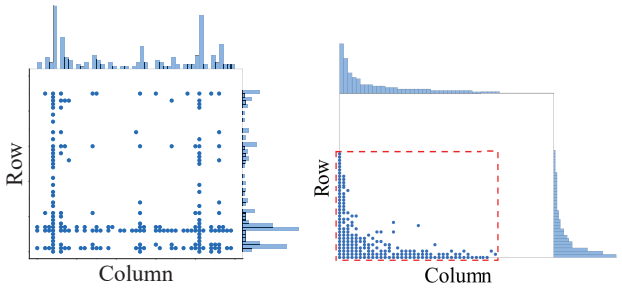
Tianran Chen[1], Jiarui Chen[1], Baoquan Zhang[1]*, Zhehao Yu[1], Shidong Chen[1], Rui Ye[2], Xutao Li[1], Yunming Ye[1]
[1]Harbin Institute of Technology, Shenzhen [2] Guizhou Power Grid Co., Ltd. Intelligent Operation Center
{23S151121, jiaruichen, 210110629, chenshidong}@stu.hit.edu.cn
{baoquanzhang, lixutao, yeyunming}@hit.edu.cn, {yeruigzyj}@163.com

Abstract

*Parameter-Efficient Fine-Tuning (PEFT) is a fundamental research problem in computer vision, which aims to tune a few parameters for efficient storage and adaptation of pre-trained vision models. Recently, sensitivity-aware parameter efficient fine-tuning method (SPT) addresses this problem by identifying sensitive parameters and then leveraging its sparse characteristic to combine unstructured and structured tuning for PEFT. However, existing methods only focus on the sparse characteristic of sensitive parameters but overlook its distribution characteristic, which results in additional storage burden and limited performance improvement. In this paper, we find that the distribution of sensitive parameters is not chaotic, but concentrates on a small number of rows or columns in each parameter matrix. Inspired by this fact, we propose a **C**ompact **D**ynamic-**R**ank **A**daptation-based tuning method for **S**ensitivity-aware **P**arameter efficient fine-**T**uning, called CDRA-SPT. Specifically, we first identify the sensitive parameters that require tuning for each downstream task. Then, we reorganize the sensitive parameters by following its row and column into a compact sub-parameter matrix. Finally, a dynamic-rank adaptation is designed and applied at sub-parameter matrix level for PEFT. Its advantage is that the dynamic-rank characteristic of sub-parameter matrix can be fully exploited for PEFT. Extensive experiments show that our method achieves superior performance over previous state-of-the-art methods.*

1. Introduction

With the development of large vision models on representation and generation, the paradigm of pre-training and fine-tuning has attracted increasing interest on computer vision [5, 15, 22, 25, 41, 42]. A widely used fine-tuning technique is the full fine-tuning, which leverages the pre-trained weights as the initial parameters and then tunes all parameters for adaptation of downstream tasks. However, as the number of model parameters increases, the full fine-tuning method needs to store many parameters and takes lots of computing power to tune pre-trained vision models for each downstream task. This results in these method are difficult to apply to resource-limited scenarios. To address this problem, parameter-efficient fine-tuning (PEFT) is proposed and has received wide attention recently. PEFT, as a fundamental problem in computer vision, aims to achieve an efficient transfer from a pre-trained vision model to target downstream tasks by freezing most of the pre-trained parameters and only tuning a few model parameters [29, 31].



(a) Original Parameters. (b) Reorganized Parameters.

Figure 1. Distribution illustration of of sensitive parameter. In fact, the distribution of sensitive parameters is not chaotic on origin parameter but concentrates in a small number of rows or columns (a). After reorganizing its row and column, these sensitive parameters will be clustered into a compact sub-parameter matrix (b). Inspired by this fact, we propose to apply structured tuning at sub-parameter matrix levels instead of entire matrix. Its advantage is the distribution character of sensitive parameters concentrating in a small of row and column can be finely leveraged for PEFT.

Existing studies [4, 6, 31, 43, 48] effectively address this problem by selecting a subset of parameters (called unstructured tuning) [4, 6, 31, 43, 48] or adding a few additional parameters (called structured tuning) [4, 6, 31, 43, 48] to perform adaptation of downstream tasks. Although these methods are all different, He *et al.* [14] recently found that these unstructured and structured tuning methods can combine and then present sensitivity-aware parameter efficient fine-tuning methods (called SPT [14]) for enhancing PEFT

*Corresponding author.

performance. Their main idea is to identify sensitive parameters and then tune these parameters in a manual combination manner between unstructured and structured tuning [14]. Such a design can fully exploit the sparse characteristic of sensitive parameters, such that the finetuning performance can obtain significant improvement.

In this paper, we also focus on sensitivity-aware efficient fine-tuning, but argue the distribution characteristic of sensitive parameters is not fully explored and utilized in existing SPT [14]. This is because existing SPT [14] methods evaluate sensitivity at the level of the entire parameter matrix (see Figure 1a), which only focuses on the sparse characteristic of sensitive parameters, but overlooks its distribution characteristic at the row and column vector levels. This results in additional storage burden and limited performance improvement. In fact, as shown in Figure 1a, we find that the distribution of sensitive parameters is not chaotic on entire parameter matrix but concentrates in a small number of rows or columns in the parameter matrix. Based on this fact, our intuitive idea is *can we further improve the efficiency of SPT based on such distribution characteristics?* To this end, we propose a novel compact dynamic-rank adaptation-based tuning method for sensitivity-aware efficient fine-tuning, which reorganizes the sensitive row and column parameters as a compact sub-parameter matrix (see Figure 1b) and designs a compact dynamic-rank adaptation for SPT, called CDRA-SPT. Specifically, we first quickly identifies and organizes the sensitive row and column parameters as a compact sub-parameter matrix. After that, we design a dynamic rank allocation strategy for sub-parameter matrix adaptation. Its advantage is that the dynamic-rank characteristic of sub-parameter matrix can be fully exploited for enhancing fine-tuning performance on downstream tasks. Our main contributions can be summarized as follows:

- We find a new distribution characteristic of sensitive parameters overlooked in previous existing studies, *i.e.*, the distribution of sensitive parameters is actually not chaotic on the entire parameter matrix, but concentrates in a small number of rows or columns in the parameter matrix.
- Resorting such distribution characteristic, we design a novel compact dynamic-rank adaptation-based tuning method to replace the combination fine-tuning between unstructured and structured tuning. Its advantage is that the relationship and sparse distribution between parameters can be fully leveraged to enhance PEFT.
- We conduct various experiments on various datasets, which verify the effectiveness of our CDRA-SFT method.

2. Related Works

2.1. Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning [13, 16, 19, 44] (PEFT) is an important machine learning method, which aims to tune only a tiny portion of parameters to achieve adaptation of pre-trained models to downstream tasks. Depending on the difference of parameter types, existing PEFT methods can be divided into three types. ***1) Addition-based PEFT Approaches.*** This type of approach attaches additional trainable parameters to the backbone and only tunes these parameters for adaptation of pre-trained models [16, 19]. For example, in [19], Jia et al. propose introducing learnable prompts as additional trainable parameters (called VPT) and then freeze the pre-trained weights to perform PEFT in a prompt learning manner. ***2) Reparameterization-based PEFT Approaches.*** This line of methods aims to avoid extra computational costs by tuning parameters that are inherently in or can be reparameterized into the backbone during inference [9, 17, 26, 45]. For instance, in [17], Hu et al. propose a low-rank adaptation (called LoRA) for PEFT, which freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of Transformer for achieving adaptation of pre-trained models. With the superiority of LoRA, various LoRA variants are proposed recently, such as DoRA [26], VeRA [23], AdaLoRA [45], LoRA+ [13], FourierFT [9], and HydraLoRA [30]. ***3) Selection-based PEFT Approaches.*** This methods focus on how select a subset from pretrained parameters and tuning them in a masked finetuning manner for adaptation of pretrained models [14, 37, 47]. In early works, existing studies mainly focus on parameter types to select parameter subset, *e.g.*, selecting bias [37] or last k-layer [16, 19] parameters as parameter subset to tune. Recently, He et al. turn into the perspective of parameter sensitivity to select subset of parameters. For instance, in [14], Hu et al. propose to select a subset of parameters from the perspective of sensitivity and then perform PEFT in a combination manner between unstructured and structured tuning, called SPT. Unlike SPT, Zhang et al. [47] perform PEFT by fining sensitive parameters in a mask-tuning manner.

In this paper, we mainly focus on selection-based PEFT approaches due to its superiority. However, different from these existing methods, 1) we find a new distribution characteristic of sensitive parameters overlooked in previous existing studies, *i.e.*, the distribution of sensitive parameters is actually not chaotic on entire parameter matrix, but concentrates in a small number of rows or columns in the parameter matrix; and 2) we present a novel compact dynamic-rank adaptation for enhancing sensitivity-aware PEFT methods.

2.2. Sub-Networks Learning

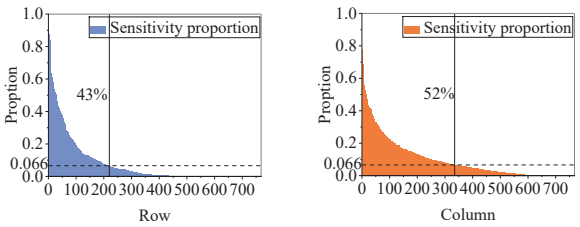
Sub-network learning [2, 8] is an important model pruning techniques, which aims to identify an important sub-network from a big model meanwhile remains superiority performance similar to the origin big model. In [8], Frankle et al. propose a lottery ticket hypothesis, which verifies that small sub-networks from big models can achieve the same level of accuracy as the original model. After that, with such a lottery ticket hypothesis theory, various sub-networks learning methods are proposed [12, 35]. For example, in [12], Guo et al. design a SpotTune method that employs a policy network to make routing decisions for sub-networks. Similarly, Xu et al. [35] develop a Child-Tuning method, which iteratively updates a subset of parameters by masking certain gradients during the backpropagation process for sub-network learning. With the superiority of sub-network learning, some studies have explored sub-network learning to various domains, such as continual learning [11, 20, 24, 36], multi-task learning [1, 18, 27, 30] and few-shot learning [3, 36, 39, 40]. Recently, several PEFT techniques relying on sub-network learning have been proposed. In [14], a sensitivity-aware parameter efficient fine-tuning method (SPT) is developed based on weight sensitivity, which aims to only finetune a sub-set of sensitive parameters to achieve PEFT. Zhang et al. [47] design a gradient-based parameter selection method (GPS) to select sub-network and then perform PEFT by mask-tuning.

In this paper, we also focus on exploring sub-networks learning to enhance PEFT. However, different from existing methods that performs PEFT by using mask-tuning manner, we present a novel compact dynamic-rank adaptation method to explore sub-networks learning for PEFT. Its advantage is the distribution character of sensitive sub-networks can be finely characterized for PEFT.

3. Methodology

3.1. Preliminaries on SPT

SPT [14], as a pioneering work on sensitivity-aware parameter efficient fine-tuning, aims to explore where to introduce and how to allocate trainable parameters for a pre-trained models in a sensitivity-aware manner. Formally, let $\{w_0, ..., w_i, ..., w_N\}$ denotes the pre-trained model weights where N is the number of parameters, and $\mathcal{D}^{train}$ denotes the training set of a given downstream task. The SPT performs the PEFT of downstream by the following two steps: (1) evaluating the sensitivity s_i of each parameter w_i by calculating the square of its accumulation gradient g_i around 400 training samples from $\mathcal{D}^{train}$, *i.e.*, $s_i = g_i^2$, and then selecting the top-γ most sensitive weight as the sensitive parameters; and (2) performing the PEFT of downstream tasks in an adaptive combination between unstructured and structured tuning, *i.e.*, using unstructured tuning when the



(a) Sparsity analysis at row (b) Sparsity analysis at column

Figure 2. Sparsity analysis of sensitive parameter.

sensitive weight matrices that have a small number of sensitive parameters otherwise structured tuning is employed. Finally, the adapted parameter w_i' can be expressed as:

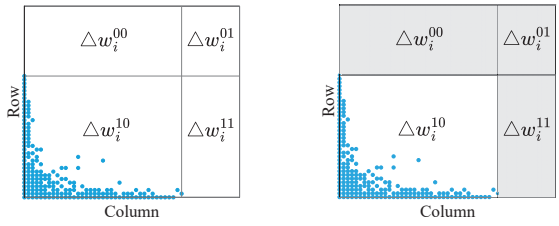
$$w_i' = \begin{cases} w_i - \eta g_{w_i} \odot m_i, & if \sum_i m_i < \alpha, \\ w_i + w_{down} w_{up}, & otherwise. \end{cases} \quad (1)$$

where η is a learning rate, m_i denotes a binary mask marking sensitive parameters, g_{w_i} is the gradient of parameter w_i, $w_{down} w_{up}$ denotes the low-rank adaptation (LoRA) [17], and α is the number of parameters of w_{down} and w_{up}.

3.2. Motivation Analysis on SPT

Although the existing SPT has shown superior performance, it only focuses on the level of the entire parameter matrix to evaluate the sparse characteristic of sensitive parameters (see Eq. 1), which overlooks its distribution characteristic at the level of row and column vectors. This results in its additional storage burden and limited performance improvement. Next, we conduct a detailed analysis on this point.

Case 1: Parameter Matrix w_i Satisfying $\sum_i m_i < \alpha$. As shown in Eq. 1, existing SPT employs a full fine-tuning with sensitivity mask m_i to tune such parameters, which is because the sensitive parameters are very sparse. However, in this paper, we find that these sensitive parameters actually are not sparse. The reason it's considered sparse is that existing SPT only focuses on the level of entire parameter matrix to evaluate its sparsity, *i.e.*, comparing the number of sensitive parameters and LoRA parameters of entire parameter matrix. However, these sensitive parameters will become dense when we evaluate its sparsity from the perspective of row and column. This is because the distribution of sensitive parameters is actually not chaotic on entire parameter matrix but concentrates in a small number of rows or columns. To illustrate this, in Figure 2, we randomly select a query parameter matrix w_i from pre-trained ViT weights and then show its sparsity (*i.e.*, the number of sensitive parameters divided by the total number of parameters) for each row (see Figure 2a), column (see Figure 2b), and entire matrix (see dotted lines of Figure 2). From results, we can see that the sensitive parameters at some row and column (around 43% and 52%) are more dense than entire matrix. This inspires us an intuitive idea that can we reor-



(a) Existing SPT method. (b) Our method.

Figure 3. An example illustration of weight change.

ganize the dense row and column parameters as a compact sub-parameter matrix and then tune the sub-parameter in a low-rank adaptation manner. Its advantage are: (1) using the low-rank adaptation to model the weight change of sensitive parameters can further reduce trainable parameters; and (2) the correlation between sensitive row and column vectors can be fully characterized for enhancing SPT.

Case 2: Parameter Matrix w_i Satisfying $\sum_i m_i \geq \alpha$. As shown in Eq. 1, to preserve efficient storage and superior performance, existing SPT employs a structured tuning (*e.g.* LoRA [17]) to replace unstructured tuning for tuning these parameters. Although such method can ensure its accuracy and efficiency, it also introduces lots of redundant parameters to achieve adaptation of non-sensitive parameters. To illustrate this issue, given a parameter w_i, we assume the matrix shown in Figure 3a is its sensitive parameter distribution. For tuning such parameters, LoRA employs two low-rank matrices A and B to model its change, *i.e.*, $\triangle w_i = AB$ with same size as w_i. As shown in Figure 3a, the weight change actually can divided into four sub-matrices ($\triangle w_i^{00}$, $\triangle w_i^{01}$, $\triangle w_i^{10}$, and $\triangle w_i^{11}$). That is,

$$\triangle w_i = AB = \left[\begin{array}{c|c} \triangle w_i^{00} & \triangle w_i^{01} \\ \hline \triangle w_i^{10} & \triangle w_i^{11} \end{array}\right]. \tag{2}$$

In fact, among them, only the low-rank sub-matrix $\triangle w_i^{10}$ accounts for modeling the weight change of sensitive row and column parameters, and others low-rank sub-matrixs (*i.e.*, $\triangle w_i^{00}$, $\triangle w_i^{01}$, and $\triangle w_i^{11}$) models either non-sensitive row parameters or non-sensitive column parameters. This means that the three low-rank sub-matrixs (*i.e.*, $\triangle w_i^{00}$, $\triangle w_i^{01}$, and $\triangle w_i^{11}$) actually are non-sensitive for downstream tasks. Thus, as shown in Figure 3b, an intuitive idea that whether can we only use the the low-rank sub-matrix $\triangle w_i^{10}$ to tune the parameter w_i? The advantage of such design is that the trainable parameter will be significantly reduce when we remove $\triangle w_i^{00}$, $\triangle w_i^{01}$, and $\triangle w_i^{11}$ such that the over-fitting issue can also be effectively alleviated.

3.3. The Proposed CDRA-SPT

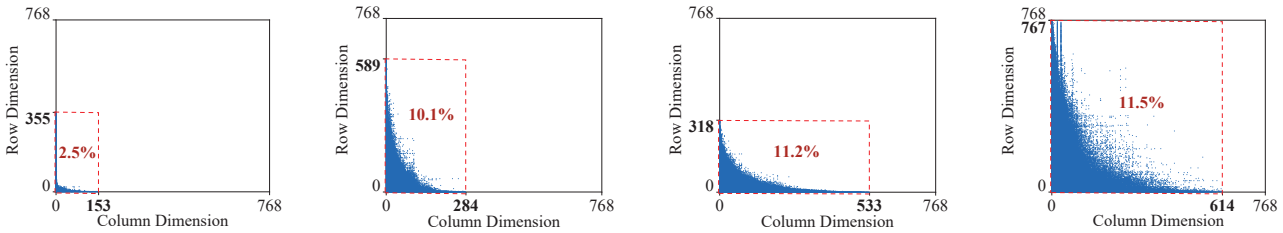
Based on the above motivation analysis, we find that the above unstructured tuning (*i.e.*, **Case 1** described in Section 3.2) and structured tuning (*i.e.*, **Case 2** described in Section 3.2) used in SPT [14] can all be unified to a structured tuning applied at the compact sub-parameter matrix level. Based on this fact, we propose a novel compact dynamic-rank adaptation-based tuning method for SPT, called CDRA-SPT. The main idea is that with the row/column distribution characteristic of sensitive parameters, we reorganize the sensitive parameters by following its row and column into a compact sub-parameter matrix, and then perform unified structured tuning at sub-parameter matrix level to enhance existing SPT performance.

Algorithm 1 PyTorch-style pseudocode for **CDRA-SPT**.

```
class CDRA_SPT(nn.Module):
    def __init__(self, m_i, in_d, out_d, r):
        super(CDRA_SPT, self).__init__()
        self.bar_d, self.bar_k, self.c_map, self.r_map =
        \self.reorganize(self.mask_w, self.d, self.k)

        self.d = in_d
        self.k = out_d
        self.lora_rank = lora_r

        self.w = nn.Parameter(torch.randn(self.d, self.k))
        self.bias = nn.Parameter(torch.randn(1, self.k))
        self.A = nn.Parameter(torch.randn(self.bar_d, lora_r))
        self.B = nn.Parameter(torch.randn(lora_r, self.bar_k))
        # binary mask matrix obtained in Step 1
        self.mask_w = m_i

    def reorganize(self, mask_w, in_dim, out_dim):
        sensitive_row = torch.sum(mask_w, dim=1)
        _, r_map = torch.sort(sensitive_row, descending=True)
        sensitive_col = torch.sum(mask_w[r_map], dim=0)
        _, c_map = torch.sort(sensitive_col, descending=True)

        # Calculate the dimension of compact matrix
        row_dim = in_dim
        col_dim = out_dim
        for i in range(in_dim):
            if torch.sum(mask_w[r_map[i], :]) == 0:
                row_dim = i
                break
        for i in range(out_dim):
            if torch.sum(mask_w[:, c_map[i]]) == 0:
                col_dim = i
                break
        return col_dim, row_dim, c_map, r_map

    def forward(self, x):
        # Reorganize in Step2
        bar_w = self.w[self.r_map, :][:, self.c_map]

        delta_w = self.A @ self.B
        # Eq 6 in Step 3
        bar_w[:self.bar_d,:][:, self.bar_k] += delta_w

        w = torch.zeros_like(self.w)
        # Reverse the reorganization in Step 3
        w[self.r_map, :][:, self.c_map] = bar_w

        return x @ w + self.bias
```

As shown in Algorithm 1, given a pre-trained model $\{w_0, ..., w_i, ..., w_N\}$ and a downstream task $\tau = \{\mathcal{D}_{tr}, \mathcal{D}_{te}\}$ where $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$ denote its training and test set, our CDRA-SPT achieve PEFT by three steps:

Step 1. we first evaluate the sensitivity s_i for each parameter $w_i \in \{w_0, ..., w_i, ..., w_N\}$. Here, the sensitivity evaluation manner is similar to SPT, *i.e.* calculating and regarding its accumulated square gradient from all training samples of downstream tasks as parameter sensitivity (*i.e.*,



(a) Query weight matrix at layer 11. (b) FFN weight matrix at layer 11. (c) FFN weight matrix at layer 10. (d) Value weight matrix at layer 6.

Figure 4. Illustration of compact sub-parameter matrices. Here, the red box represents the transformed compact sub-parameter matrices.

$s_i = g_i^2$ where g_i is its gradient). That is,

$$g_i^2 = \frac{\sum_{x \in D^{tr}} (\nabla \mathcal{L}_x(w_i)^2)}{\|D^{tr}\|}, s_i = g_i^2, \tag{3}$$

where $\|\|$ is the set size and $\nabla \mathcal{L}_x(w_i)$ denotes the gradients from training sample x. Finally, we select the top-γ sensitive weight as the sensitive parameters. In particular, we denote these sensitive parameters for each parameter matrix w_i by using a binary mask m_i where "1" denotes sensitive parameters and "0" denotes non-sensitive parameters.

Step 2. To fully exploit the distribution character of sensitive parameters, we propose to reorganize each parameter matrix w_i in a descending order of importance along its rows and columns. As a result, a new parameter $\bar{w}_i$ can be obtained where these sensitive parameters can be clustered into a compact sub-parameter matrix. That is,

$$\bar{w}_i = Reorganize(w_i) = \left[\begin{array}{c|c} \bar{w}_i^{00} & \bar{w}_i^{01} \\ \hline \bar{w}_i^{10} & \bar{w}_i^{11} \end{array}\right], \tag{4}$$

where $Reorganize()$ denotes parameter transformation and all sensitive parameters will be clustered to sub-parameter matrix $\bar{w}_i^{10}$. Note that the parameter transformation $Reorganize()$ is only the reordering of rows and columns, which is inverse. Thus, the original parameter w_i can be again obtained by using a reverse transformation. That is,

$$w_i = Reorganize^{-1}(\bar{w}_i), \tag{5}$$

where $Reorganize^{-1}()$ is the reverse transformation.

Step 3. Considered that all sensitive parameters are distributed in sub-parameter matrix $\bar{w}_i^{10}$, instead of performing PEFT on the entire parameter matrix, we propose to directly perform low-rank adaptation on the sub-parameter matrix $\bar{w}_i^{10}$ to model its weight change $\triangle \bar{w}_i$. As a result, an adapted parameter matrix $\bar{w}_i'$ can be obtained. That is,

$$\bar{w}_i' = \bar{w}_i + \triangle \bar{w}_i = \left[\begin{array}{c|c} \bar{w}_i^{00} & \bar{w}_i^{01} \\ \hline \bar{w}_i^{10} + AB & \bar{w}_i^{11} \end{array}\right], \tag{6}$$

where A and B denotes its learnable low-rank matrices. Finally, we employ the reverse transformation described in Eq. 5 to achieve the adapted parameters w_i' in original parameter space and then perform PEFT of downstream task.

3.4. Dynamic-Rank Allocation Strategy

Until now, we have introduced all framework details of our CDRA-SPT, a remaining problem is: how to allocate the rank for low-rank matrices $A \in \mathcal{R}^{d^i \times r}$ and $B \in \mathcal{R}^{r \times k^i}$ used to model the weight change of sub-parameter matrix $\bar{w}_i^{10}$. A direct and simple allocation method is following existing low-rank adaptation methods [17] and setting fixed rank (*e.g.*, $r = 8$ or $r = 16$) for modeling all sub-parameter matrix $\bar{w}_i^{10}$. The assumption behind such approach is that the intrinsic dimension of all sub-parameter matrix $\bar{w}_i^{10}$ from pretrained models are all the same. However, a nature question is "*Is this assumption reasonable?*"

To answer this question, in Figure 4, we randomly select four parameter matrices from various parameter types (*i.e.*, query matrix, value matrix, and feedforward matrix from transformer layers) and visualize the sensitive parameters of reorganized compact parameter matrices (*i.e.*, the parameter matrix $\bar{w}_i$ obtained by Eq. 4). The results are shown in Figure 4. From the results, we can see that although the size of parameter matrix $w_i \in \{w_0, ..., w_i, ..., w_N\}$ are all the same (*i.e.*, 768×768), its size of compact sub-parameter matrices $\bar{w}_i^{10}$ is different. For example, the size of compact sub-parameter matrices $\bar{w}_i^{10}$ from 15-layer FNN weight matrix is 589×284, but the ones from 10-layer FNN weight matrix is 318×533. Besides, we also calculate the proportion of the sensitive parameters in the compact sub-parameter matrices $\bar{w}_i^{10}$ (see the red number of Figures. 5a $\sim$ 5e). From results, we can see that the proportion of the sensitive parameters are all different. The above results suggest that setting fixed rank (*e.g.*, $r = 8$ or $r = 16$) for modeling all sub-parameter matrix change is difficult to fit its change of size and proportion of the sensitive parameters of sub-parameter matrices.

Based on the above analysis, an intuitive idea is *how allocate the matrix rank in an adaptive manner such that the dynamic characteristics of compact sub-parameter matrices $\bar{w}_i^{10}$ can be finely modeled*. Based on this, we propose a novel dynamic-rank allocation strategy. Our key insight is that the number of sensitive parameters and the compact matrix's dimension may reflect the the intrinsic dimension of compact sub-parameter matrix $\bar{w}_i^{10}$, *i.e.*, (1) the smaller the number of sensitive parameters, the smaller the intrinsic dimension and (2) the smaller the dimensions of a compact matrix, the smaller the intrinsic dimension. Based on the

Method \ Dataset	Natural								Specialized					Structured									VTAB	
	CIFAR-100	Caltech101	DTD	Flowers102	Pets	SVHN	Sun397	Mean Acc.	Patch Camelyon	EuroSAT	Resisc45	Retinopathy	Mean Acc.	Clevr/count	Clevr/distance	DMLab	KITTI/distance	dSprites/loc	dSprites/ori	SmallNORB/azi	SmallNORB/ele	Mean Acc.	Mean Acc.	Mean Params. (%)
Full [19]	68.9	87.7	64.3	97.2	86.9	87.4	38.8	75.9	79.7	95.7	84.2	73.9	83.4	56.3	58.6	41.7	65.5	57.5	46.7	25.7	29.1	47.6	65.57	100.00
Addition-based methods																								
MLP-3 [19]	63.8	84.7	62.3	97.4	84.7	32.5	49.2	67.8	77.0	88.0	70.2	56.1	72.8	47.8	32.8	32.3	58.1	12.9	21.2	15.2	24.8	30.6	53.21	1.50
VPT-Shallow [19]	77.7	86.9	62.6	97.5	87.3	74.5	51.2	76.8	78.2	92.0	75.6	72.9	79.7	50.5	58.6	40.5	67.1	68.7	36.1	20.2	34.1	47.0	64.85	0.13
VPT-Deep [19]	78.8	90.8	65.8	98.0	88.3	78.1	49.6	78.5	81.8	96.1	83.4	68.4	82.4	68.5	60.0	46.5	72.8	73.6	47.9	32.9	37.8	55.0	69.43	0.70
Adapter [16]	69.2	90.1	68.0	98.8	89.9	82.8	54.3	79.0	84.0	94.9	81.9	75.5	84.1	80.9	65.3	48.6	78.3	74.8	48.5	29.9	41.6	58.5	71.44	0.39
NOAH [46]	69.6	92.7	70.2	99.1	90.4	86.1	53.7	80.2	84.4	95.4	83.9	75.8	84.9	82.8	68.9	49.9	81.7	81.8	48.3	32.8	44.2	61.3	73.25	0.50
Reparameterization-based methods																								
Linear [19]	63.4	85.0	64.3	97.0	86.3	36.6	51.0	69.1	78.5	87.5	68.6	74.0	77.2	34.3	30.6	33.2	55.4	12.5	20.0	9.6	19.2	26.9	53.00	0.05
PARTIAL-1 [19]	66.8	85.9	62.5	97.3	85.5	37.6	50.6	69.4	78.6	89.8	72.5	73.3	78.5	41.5	34.3	33.9	61.0	31.3	32.8	16.3	22.4	34.2	56.52	8.38
Bias [37]	72.8	87.0	59.2	97.5	85.3	59.9	51.4	73.3	78.7	91.6	72.9	69.8	78.3	61.5	55.6	32.4	55.9	66.6	40.0	15.7	25.1	44.1	62.05	0.16
LoRA [17]	68.1	91.4	69.8	99.0	90.5	86.4	53.1	79.8	85.1	95.8	84.7	74.2	84.9	83.0	66.9	50.4	81.4	80.2	46.6	32.2	41.1	60.2	72.63	0.90
Selection-based methods																								
GPS [47]	70.8	93.9	74.8	99.4	82.4	91.4	51.6	80.6	87.2	95.7	86.1	76.1	86.3	80.9	61.8	54.0	81.4	84.2	52.6	30.2	45.5	61.3	73.68	0.25
SPT-LoRA-8 [14]	72.8	92.7	72.6	99.3	91.2	86.7	55.2	81.5	85.5	95.8	85.7	75.6	85.7	82.0	68.1	49.4	81.4	80.2	48.6	29.4	39.7	59.9	73.26	0.51
SPT-LoRA-16 [14]	71.9	93.2	71.0	99.3	91.2	89.6	55.2	81.6	86.4	96.2	85.6	76.2	86.1	83.2	67.0	52.0	82.9	81.8	49.6	32.3	39.0	61.0	73.87	1.36
CDRA-SPT (Ours)	73.8	93.3	72.7	99.4	91.6	89.9	55.8	82.4	86.8	96.2	86.2	76.0	86.3	83.0	68.9	51.5	82.1	80.9	51.7	33.0	43.2	61.8	**74.53**	0.49

Table 1. Performance comparisons on VTAB-1k with ViT-B/16 models pre-trained on ImageNet-21K. Best results are highlighted in bold.

above insight, we propose dynamically allocating compact matrix's rank $\bar{r^i}$ according to the sensitive parameters and compact matrix's dimension (*i.e.*, its row $\bar{d^i}$ and column $\bar{k^i}$). That is, finding a maximum rank r as $\bar{r^i}$ and satisfy:

$$\begin{aligned} \bar{r^i} &= \max(r), \\ \textbf{s.t.} \sum m_i &\geq (\bar{d^i} \times r + \bar{k^i} \times r) \times \frac{d^i \times k^i}{\bar{d^i} \times \bar{k^i}}), \end{aligned} \tag{7}$$

where d^i and k^i denote the dimension of the original matrix, and m_i denotes the binary mask where its value '1' denotes the corresponding weights are sensitivity parameters. The goal of Eq. 7 is to ensure that the number of trainable parameters of our CDRA is always lower than the number of sensitive parameters, and the smaller the dimension of compact sub-parameter matrix $\bar{w}_i^{10}$, the lower its number.

4. Experiments

4.1. Experimental Settings

Datasets. Following SPT [14], we evaluate our method of CDRA-SPT and various baselines on a series of datasets, including image classification and semantic segmentation tasks. For image classification, we use: **VTAB-1k:** [38] Visual Task Adaptation Benchmark (VTAB), which consists of 19 visual classification tasks grouped into three types: natural image classification tasks, specialized image classification tasks, and structured image classification tasks. **FGVC:** Fine-Grained Visual Classification (FGVC) benchmark, which includes five downstream tasks: CUB-200-2011 [34], NABirds [33], Oxford Flowers [28], Stanford-Dogs [21], and StanfordCars [10].

For semantic segmentation, we evaluate on: **ADE20K:** [49]A large-scale dataset, with over 20,000 images annotated across 150 object and stuff categories, is commonly used for semantic segmentation tasks due to its diverse scenes and detailed annotations. **VOC:** [7] The PASCAL VOC dataset, consisting of 20 object categories with pixel-level annotations, is known for its well-defined evaluation protocols and balanced class distribution.

Baselines and Evaluation Metrics. In order to demonstrate the effectiveness of our CDRA-SPT method, we compare our CDRA-SPT with four type parameter-efficient fine-tuning methods: (i) **Full**: As the most commonly used method, full fine-tuning updates all parameters of the whole model for adaptation to downstream tasks. (ii) **Addition-based**: The trainable parameters are added to the model in this method, but require extra computational cost during inference. Consisting of MLP-k, Adapter [16], VPT [19] and NOAH [46]. (iii) **Reparameterization-based**: The trainable parameters can be merged into the original parameter matrix after fine-tuning without computational cost during inference, including LINEAR, PARTIAL-k, BIAS [37], and LORA [17]. (iv)**Selection-based**: select some important parameters as a subset of parameters in the backbone for fine-tuning, including SPT [14] and GPS [47]. In particular, our CDRA-SPT method can be categorized into the type of selection-based PEFT approaches, thus existing SPT [14] and GPS [47] methods are our key competitors in this paper.

Implementation Details. Following SPT [14], we employ AdamW optimizer with cosine learning rate decay to fine-tune the model for 100 epochs. And we use the standard data augmentation pipeline to process images in all datasets of the FGVC, VTAB-1k. We set the same number of training samples as the SPT to select sensitive parameters. All experiments are conducted on NVIDIA 4090 GPU.

Dataset	CUB-2011	NABrids	OxfordFlowers	Stan.Dogs	Stan.Cars	MeanAcc.	Params.(%)
Full [19]	87.3	82.7	98.8	89.4	84.5	88.54	100.00
Linear [19]	85.3	75.9	97.9	86.2	51.3	79.32	0.21
Bias [37]	88.4	84.2	98.8	91.2	79.4	88.40	0.33
Adapter [16]	87.1	84.3	98.5	89.8	68.6	85.66	0.48
LoRA [17]	85.6	79.8	98.9	87.6	72.0	84.78	0.90
VPT-Shallow [19]	86.7	78.8	98.4	90.7	68.7	84.62	0.29
VPT-Deep [19]	88.5	84.2	99.0	90.2	83.6	89.11	0.99
SPT-LoRA-8 [14]	88.0	83.5	99.3	89.8	86.7	89.46	1.20
SPT-LoRA-16 [14]	88.2	83.8	99.3	89.7	87.4	89.69	2.28
CDRA-SPT (Ours)	88.6	83.4	99.3	89.8	88.2	**89.86**	0.86

Table 2. Results of image classification on FGVC datasets.

Dataset	ADE20k			VOC		
Method	**mDice (↑)**	**mIoU (↑)**	**Params.(%)**	**mDice (↑)**	**mIoU (↑)**	**Params.(%)**
Bias	72.8	60.7	0.12	74.3	62.4	0.12
LoRA	73.8	61.9	1.20	75.0	63.9	0.34
Adapter	72.9	60.9	0.34	76.7	65.9	0.34
SPT-LoRA-8	73.3	61.4	0.83	76.5	65.2	0.87
SPT-LoRA-16	73.3	61.4	1.60	76.9	66.3	1.70
CDRA (ours)	**74.0**	**62.0**	0.82	**77.4**	**66.4**	0.70

Table 3. Result of semantic segmentation on ADE20K and VOC.

4.2. Discussion of Results

Image Classification Tasks. We evaluate various baselines and our CDRA-SPT on the FGVC and VTAB benchmarks. The results are reported in Tables 1 and 2, respectively. From the results, it can be found that our CDRA-SPT outperforms all baseline methods on both FGVC and VTAB, which sufficiently proves the effectiveness of our method. Specifically, on VTAB dataset, we can see that 1) our CDRA-SPT exceeds existing addition-based methods by a large margin. This is reasonable since our CDRA-SPT explores weight sensitivity, which can finely characterize distribution of downstream task; 2) different from reparameterization-based methods, our CDRA-SPT focuses on selecting sensitive sub-parameters to perform PEFT of downstream tasks. This improvement results verify the superiority of our CDRA-SPT; and 3) our CDRA-SPT obtains around 1% $\sim$ 2% accuracy improvement of the mean accuracy compared to selection-based methods. In particular, our CDRA-SPT method also exceeds our key baseline (*i.e.*, SPT-LoRA-8 even SPT-LoRA-16), while it only use 0.49% of trainable parameters and is below the parameter number of SPT-LoRA-8, which verifies our CDRA-SPT superiority. Besides, on FGVC, we also similar performance improvements, *i.e.*, 1) our CDRA-SPT also achieves best classification over previous PEFT methods and 2) our CDRA-SPT exceeds our key competitors (*i.e.*, SPT) by around 2 %. This further shows the superiority of our CDRA-SPT.

Semantic Segmentation Tasks. In Table 3, we conduct semantic segmentation experiments on ADE20K[49] and VOC [7]datasets to show the universality of our CDRA-SPT. In particular, following [14], 1) we select DINOV2 [32] as our pre-trained models and report mDice and mIoU as our comparison metrics; and 2) we implement Bias [37], LoRA [17], Adapter [16], SPT-LoRA-8, and SPT-LoRA-16 as our baselines. From these results, we can see that 1) our CDRA-SPT achieves the best segmentation performance on all datasets. This further verifies the effectiveness; and 2) our CDRA-SPT only use around 0.4% trainable parameters, which is below the number of our key baseline (*i.e.*, SPT-LoRA-8, and SPT-LoRA-16). This means that our CDRA-SPT indeed alleviates the parameter redundancy issue of existing SPT while achieving better PEFT performance.

	Method	Natural	Specialized	Structured	Mean Acc	Params. (%)
(i)	Dynamic rank	82.4	86.3	61.8	76.8	0.49
(ii)	Fixed Rank-8	81.0	84.5	59.4	75.0	0.42
(iii)	Fixed Rank-16	81.0	85.2	59.2	75.1	0.76
(iv)	Fixed Rank-32	80.8	85.2	60.1	75.4	1.45

Table 4. Analysis of compact dynamic-rank adaptation on VTAB.

	Case1	Case2	Natural	Specialized	Structured	Mean Acc	Params. (%)
(i)			80.9	85.3	60.0	75.4	0.94
(ii)		✓	81.2	85.5	60.3	75.7	0.88
(iii)	✓		81.4	85.8	60.2	75.8	0.46
(iv)	✓	✓	81.7	85.8	60.3	76.0	0.37

Table 5. Ablation study of our CDRA on Cases 1 and 2 of Eq. 1.

4.3. Ablation Study

Is our idea (*i.e.*, applying dynamic-rank adaptation at compact sub-parameter matrix) effective? In Table 4, we conduct an ablation study on VTAB dataset to show the effectiveness of introducing dynamic rank adaptation at compact sub-parameter matrix. Specially, (i) we implement our CDRA-SPT as our baseline (*i.e.* applying dynamic-rank adaptation at compact sub-parameter matrix); Then, on (ii) $\sim$ (iv) we replace the dynamic-rank adaptation of our CDRA-SPT by using a fixed-rank (*i.e.*, 8-rank adaptation, 16-rank adaptation, and 32-rank adaptation) adaptation used in existing SPT method, respectively. From the results, we can see that the performance of (i) outperforms (ii) $\sim$ (iv) by a large margin and the number of trainable parameters on (i) is lower than the ones of (iii) $\sim$ (iv), which is reasonable because the dynamic-rank characteristic of sub-parameter matrix can be fully exploited for PEFT. This suggests that our idea, *i.e.*, applying dynamic-rank adaptation at compact sub-parameter matrix, is very effective for SPT.

Is our CDRA (*i.e.*, compact dynamic-rank adaptation) all effective for cases 1 and 2 of SPT? To answer this question, in Table 5, we conduct an ablation study on VTAB dataset. Specially, (i) we remove our CDRA at Cases 1 and 2 of Eq. 1 as our baseline (*i.e.* using Eq. 1 to perform PEFT); (ii) we replace the mask tuning of Eq. 1 by using our CDRA on (i), *i.e.*, applying our CDRA at Case 1 of Eq. 1; (iii) we replace the low-rank adaptation of Eq. 1 by using our CDRA on (i), *i.e.*, applying our CDRA at Case 2 of Eq. 1; and (iv) we apply our CDRA at both Case 1 and Case 2 of Eq. 1 on (i), which is exactly our CDRA-SPT method. From these results, we can see that 1) the performance of (ii) $\sim$ (iv) exceed (i) by around 0.1 % $\sim$ 1%, which suggests that our CDRA is very effective, which can alleviate parameter redundancy meanwhile achieving performance improvement; and 2) the setting of (iv) achieves best PEFT performance and minimum number of trainable parameters,

	Method	Natural	Specialized	Structured	Mean Acc	Params. (%)
(i)	Baseline (SPT)	81.5	85.7	59.9	75.7	0.51
(ii)	+ CDRA-SPT ($\times$1.0)	82.4	86.3	61.8	76.8	0.49
(iii)	+ CDRA-SPT ($\times$1.5)	81.0	85.0	58.9	75.0	0.54
(iv)	+ CDRA-SPT ($\times$0.5)	81.0	84.9	57.9	74.6	0.29

Table 6. Analysis of sub-parameter matrix's size on VTAB.

	Method	Natural	Specialized	Structured	Mean Acc	Params. (%)
(i)	Baseline (SPT)	81.5	85.7	59.9	75.7	0.51
(ii)	+ CDRA ($\triangle w_i^{00}$)	81.2	85.3	59.2	75.2	0.74
(iii)	+ CDRA ($\triangle w_i^{01}$)	80.7	85.3	58.4	74.8	0.48
(iv)	+ CDRA ($\triangle w_i^{10}$)	82.4	86.3	61.8	76.8	0.49
(v)	+ CDRA ($\triangle w_i^{11}$)	81.2	84.6	58.9	74.9	0.73

Table 7. Analysis of different compact sub-parameter matrices.

Method	Inference Time (ms/img)	Inference Memory (GB)	Fine-tuning Memory (GB)
Full	2.8	1.3	11.9
VPT-Deep	3.8	1.9	13.2
LoRA	2.8	1.3	8.2
SPT	2.8	1.3	9.8
CDRA-SPT	2.8	1.3	8.3

Table 8. Anaysis of computational cost on VTAB.

which verifies the superiority of our CDRA-SPT method.

How does the size of sub-parameters impact the performance of our method? In Table 6, we analyzed the impact of the size of sub-parameters on PEFT performance. Specifically, (i) we first implement existing SPT as our baseline; Then, (ii) we apply our CDRA on (i), which is exactly our CDRA-SPT; (iii) we expand the size of compact sub-parameter matrix on (ii) to 1.5 times; (iv) we reduce the size of compact sub-parameter matrix on (ii) to 0.5 times. From these results, we can see that 1) our CDRA-SPT achieves the best PEFT performance; and 2) the performance of our CDRA-SPT decreases around 1% $\sim$ 2%, which may be because too large or too small sub-parameter matrix leads to overfitting or underfitting issues of downstream tasks.

How does tuning different sub-parameters impact performance? To answer this question, we achieve our CDRA-SPT on different sub-parameter matrices (*i.e.*, $\triangle w_i^{00}$, $\triangle w_i^{01}$, $\triangle w_i^{10}$, and $\triangle w_i^{11}$) on the VTAB dataset. The results are reported in Table 7. From these results, it can be observed is that our CDRA-SPT applying at sub-parameter matrices $\triangle w_i^{10}$ achieve the best PEFT performance. This is reasonable since the sub-parameter matrices $\triangle w_i^{10}$ indeed is more sensitive than sub-parameter matrices $\triangle w_i^{00}$, $\triangle w_i^{01}$, and $\triangle w_i^{11}$), after reorganizing the parameter matrix in theory. Thus, fine-tuning $\triangle w_i^{10}$ is more effective.

How much our CDRA-SPT take computational cost? In Table 8, we report the computational cost of our CDRA-SPT where we select fullfine-tuning, VPT-Deep, LORA, and SPT as our comparison baselines. From results, we observe that 1) the VPT-Deep method takes higher inference latency and GPU memory, which is because it introduces additional prompts as inputs; and 2) our CDRA-SPT achieves comparable inference latency and GPU memory with Full, LoRA, SPT methods. This is reasonable because the learned parameter (*i.e.*, weight changes) by our CDRA-SPT can also be reparameterized and merged into the pre-trained model. Thus, our CDRA-SPT does not consume additional cost to perform downstream tasks. Besides, it can be found that our CDRA-SPT takes less memory than baselines during fine-tuning, which verifies our efficiency.

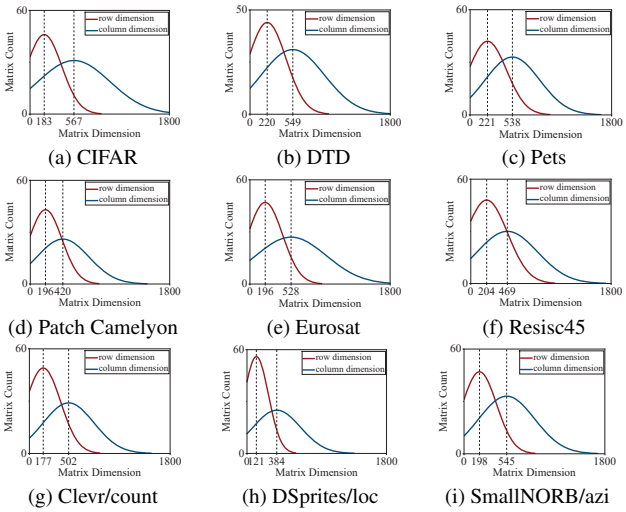


Figure 5. Distributation of the size number of compact sub-parameters at natural, specialized, and structured datasets.

How does the row/column dimension of compact sub-parameter matrix distribute at different datasets? To answer this question, in Figure 5, we visualize the dimension distribution of compact sub-parameter matrix at all layers. Specifically, we randomly select three datasets from natural image classification tasks, specialized image classification tasks, and structured image classification task, respectively, and then count the number of rows (see red line) and columns (see blue line) of all compact sub-parameter matrices. From the results, we can see that most parameter matrices all are grouped into a very compact sub-parameter matrix with mean row dimension of 121 $\sim$ 221 and mean column dimension of 384 $\sim$ 567, which verifies the rationality of our idea (*i.e.*, compact dynamic-rank adaptation).

5. Conclusions

In this paper, we find a new distribution character of sensitive parameters, *i.e.*, these sensitive parameters usually concentrate in a small number of rows or columns. Resorting to such characters, this paper proposes a novel compact dynamic-rank adaptation method for sensitivity-aware parameter efficient fine-tuning, called CDRA-SPT. Its advantage is that the dynamic-rank characteristic of sub-parameter matrix can be fully exploited for PEFT. Experimental results on various datasets show that our CDRA-SPT achieves superior performance over previous methods.

Acknowlegements

This work was supported by the Guangdong Basic and Applied Basic Research Foundation under Grant No.

2025A1515011674, the Shenzhen Peacock Program under Grant No. ZX20230597, NSFC under Grant No. 62272130 and Grant No. 62376072.

References

[1] Ahmed Agiza, Marina Neseem, and Sherief Reda. Mtlora: Low-rank adaptation approach for efficient multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16196–16205, 2024. 3

[2] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *ArXiv*, abs/2007.12223, 2020. 3

[3] Daiki Chijiwa, Shin'ya Yamaguchi, Atsutoshi Kumagai, and Yasutoshi Ida. Meta-ticket: Finding optimal subnetworks for few-shot learning within randomly initialized neural networks. *Advances in Neural Information Processing Systems*, 35:25264–25277, 2022. 3

[4] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, et al. Cogview: Mastering text-to-image generation via transformers. *Advances in Neural Information Processing Systems*, 34:19822–19835, 2021. 1

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 1

[6] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021. 1

[7] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 6, 7

[8] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018. 3

[9] Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-efficient fine-tuning with discrete fourier transform. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. 2

[10] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. Fine-grained car detection for visual census estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017. 6

[11] Dipam Goswami, Albin Soutif-Cormerais, Yuyang Liu, Sandesh Kamath, Bart Twardowski, Joost van de Weijer, et al. Resurrecting old classes with new data for exemplar-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28525–28534, 2024. 3

[12] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4805–4814, 2019. 3

[13] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. In *Forty-first International Conference on Machine Learning*. 2

[14] Haoyu He, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. Sensitivity-aware visual parameter-efficient fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11825–11835, 2023. 1, 2, 3, 4, 6, 7

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022. 1

[16] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019. 2, 6, 7

[17] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*. 2, 3, 4, 5, 6, 7

[18] Wooseong Jeong and Kuk-Jin Yoon. Quantifying task priority for multi-task optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 363–372, 2024. 3

[19] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2022. 2, 6, 7

[20] Haeyong Kang, Rusty John Lloyd Mina, Sultan Rizky Hikmawan Madjid, Jaehong Yoon, Mark Hasegawa-Johnson, Sung Ju Hwang, and Chang D Yoo. Forget-free continual learning with winning subnetworks. In *International Conference on Machine Learning*, pages 10734–10750. PMLR, 2022. 3

[21] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR workshop on fine-grained visual categorization (FGVC)*, 2011. 6

[22] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 1

[23] Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. Vera: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*. 2

[24] Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23638–23647, 2024. 3

[25] Yan-Bo Lin, Yi-Lin Sung, Jie Lei, Mohit Bansal, and Gedas Bertasius. Vision transformers are parameter-efficient audio-visual learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2299–2309, 2023. 1

[26] Shih-yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*. 2

[27] Pierre Marza, Laetitia Matignon, Olivier Simonin, and Christian Wolf. Task-conditioned adaptation of visual features in multi-task policy learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17847–17856, 2024. 3

[28] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008. 6

[29] Jialun Peng, Dong Liu, Songcen Xu, and Houqiang Li. Generating diverse structure for image inpainting with hierarchical vq-vae. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10775–10784, 2021. 1

[30] Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 2, 3

[31] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 1

[32] Rob van Gastel. Finetuning dinov2 with lora for image segmentation. 2024. 7

[33] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 595–604, 2015. 6

[34] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 6

[35] Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning. *arXiv preprint arXiv:2109.05687*, 2021. 3

[36] Jaehong Yoon, Sultan Madjid, Sung Ju Hwang, Chang-Dong Yoo, et al. On the soft-subnetwork for few-shot class incremental learning. In *International Conference on Learning Representations (ICLR) 2023*. International Conference on Learning Representations, 2023. 3

[37] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 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 (Volume 2: Short Papers)*, pages 1–9, 2022. 2, 6, 7

[38] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019. 6

[39] Baoquan Zhang, Xutao Li, Yunming Ye, Zhichao Huang, and Lisai Zhang. Prototype completion with primitive knowledge for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3754–3762, 2021. 3

[40] Baoquan Zhang, Chuyao Luo, Demin Yu, Xutao Li, Huiwei Lin, Yunming Ye, and Bowen Zhang. Metadiff: Meta-learning with conditional diffusion for few-shot learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 16687–16695, 2024. 3

[41] Baoquan Zhang, Huaibin Wang, Chuyao Luo, Xutao Li, Guotao Liang, Yunming Ye, Xiaochen Qi, and Yao He. Codebook transfer with part-of-speech for vector-quantized image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7757–7766, 2024. 1

[42] Baoquan Zhang, Bingqi Shan, Aoxue Li, Chuyao Luo, Yunming Ye, and Zhenguo Li. Zookt: Task-adaptive knowledge transfer of model zoo for few-shot learning. *Pattern Recognition*, 158:110960, 2025. 1

[43] Jiahui Zhang, Fangneng Zhan, Christian Theobalt, and Shijian Lu. Regularized vector quantization for tokenized image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18467–18476, 2023. 1

[44] Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*, 2023. 2

[45] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. 2

[46] Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. Neural prompt search. *arXiv preprint arXiv:2206.04673*, 2022. 6

[47] Zhi Zhang, Qizhe Zhang, Zijun Gao, Renrui Zhang, Ekaterina Shutova, Shiji Zhou, and Shanghang Zhang. Gradient-based parameter selection for efficient fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28566–28577, 2024. 2, 3, 6

[48] Chuanxia Zheng and Andrea Vedaldi. Online clustered codebook. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22798–22807, 2023. 1

[49] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 6, 7