1% VS 100%: Parameter-Efficient Low Rank Adapter for Dense Predictions

Dongshuo Yin[1,2,†], Yiran Yang[1,2,†], Zhechao Wang[1,2], Hongfeng Yu[1], Kaiwen Wei[1,2], Xian Sun[1,2,*]
[1]Key Laboratory of Network Information System Technology,
Aerospace Information Research Institute, Chinese Academy of Sciences
[2]School of Electronic, Electrical and Communication Engineering,
University of Chinese Academy of Sciences
{yindongshuo19, yangyiran19, wangzhechao21, weikaiwen19}@mails.ucas.ac.cn
{yuhf, sunxian}@aircas.ac.cn

Abstract

Fine-tuning large-scale pre-trained vision models to downstream tasks is a standard technique for achieving state-of-the-art performance on computer vision benchmarks. However, fine-tuning the whole model with millions of parameters is inefficient as it requires storing a same-sized new model copy for each task. In this work, we propose LoRand, a method for fine-tuning large-scale vision models with a better trade-off between task performance and the number of trainable parameters. LoRand generates tiny adapter structures with low-rank synthesis while keeping the original backbone parameters fixed, resulting in high parameter sharing. To demonstrate LoRand's effectiveness, we implement extensive experiments on object detection, semantic segmentation, and instance segmentation tasks. By only training a small percentage (1% to 3%) of the pre-trained backbone parameters, LoRand achieves comparable performance to standard fine-tuning on COCO and ADE20K and outperforms fine-tuning in low-resource PASCAL VOC dataset.

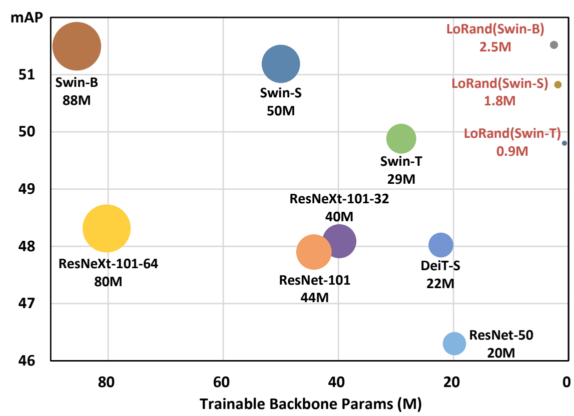


Figure 1. Comparisons of trainable backbone parameters between our methods (red) and fine-tuning (black). In COCO, we achieve advanced performances and outperform most existing backbones with only 0.9∼2.5M new backbone parameters (Cascade-RCNN is employed as the detector). The fine-tuning paradigm produces massive redundant backbone parameters, whereas our approach saves over 97% of hardware resources with competitive performances. The sizes of the circles intuitively compare the number of trainable parameters.

1. Introduction

With the rapid development of computer vision, parameters in deep models are surging. Giant models need to be trained with massive resources to achieve superior performance [3, 17, 47, 58], which is often unavailable to many academics and institutions. "Pretrain & Finetuning" paradigm is widely used to alleviate this dilemma. Teams with sufficient computation resources utilise enormous datasets [2, 9, 40, 50] to train superior backbones [4, 32, 40, 48] and optimise the models with ideal performances. Models pretrained in this way usually have a superior understanding of homogeneous data. After that, researchers with limited computational resources can transfer the understanding capabilities of the pre-trained models to downstream tasks with promising performances by fine-tuning [1, 26, 46, 53].

However, the fine-tuned model will produce a new set of parameters as large as the pre-trained model. New parameters are independent of the pre-trained models and unshareable, which are very hardware intensive for cloud service providers [23, 49]. Figure 1 compares the parameter quantities of some remarkable backbones and their performances on the COCO [28] dataset. Recent advances in natural language processing (NLP) [30, 38] show that large pre-trained models trained with rich data have strong gener-

*Corresponding author.
†Equal contribution.

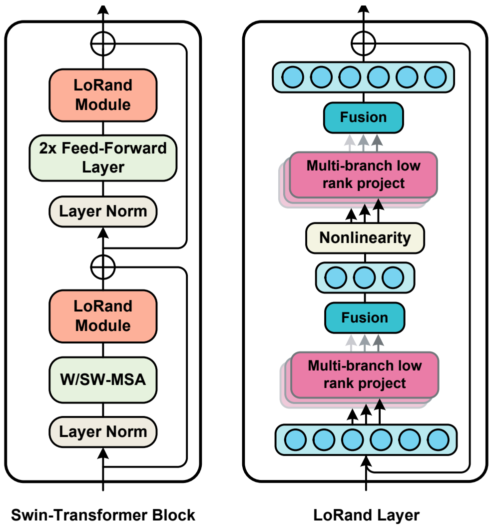


Figure 2. Architecture of the adapter module and its integration with the Transformer. **Left:** We add two LoRand structures to each SwinBlock located behind the W/SW-MSA and MLP structures respectively. **Right:** LoRand contains two Multi-branch low-rank projections and nonlinearity. We include skip-connection to LoRand to enhance its robustness.

alisability, which means most parameters in the pre-trained models can be shared with the new tasks [22, 36, 37, 44, 59]. Moreover, recent literature demonstrates that the feature understanding of pre-trained models could be reduced when they are fine-tuned in low-resource situations [12, 36]. To tackle these issues, NLP researchers propose two new training paradigms based on pre-trained models: Adapter Tuning [22] and Prompt Tuning [30], both of which tune the new models by fixing the pre-trained parameters and adding a few trainable structures (less than 10% of the backbone). These paradigms create a new buzz in NLP and achieve impressive performances which can be competitive with fine-tuning [12, 22, 30, 36–38, 44, 59]. Advances in NLP also shed new light on computer vision. Jia *et al*. [24] propose Visual Prompt Tuning (VPT) and demonstrate that VPT can outperform fine-tuning on image classification tasks by training a small number of trainable parameters. Nevertheless, VPT shows weakness on more challenging dense predictions like semantic segmentation compared with fine-tuning [24].

To find a parameter-efficient paradigm with promising performance in computer vision, we explore the potential of Adapter Tuning for visual dense predictions. We employ the advanced Swin Transformer [32] trained with ImageNet-22K [9] as the pre-trained model. After that, we add bottleneck adapter structures [22] behind each SwinBlock and freeze the original backbone parameters when training, but this approach cannot achieve comparable performance to fine-tuning as mentioned in [24]. In the experiments, we find that the models perform better with sparser adapter structures. To improve the performance of Adapter Tuning, we propose **Low-Rank Adapter (LoRand)** to reduce the adapter parameters, as shown in Figure 2. LoRand sparsely parameterizes the matrices in adapters by low-rank synthesis. Specifically, the projection matrix of the fully-connected layer (FC) in LoRand is a product of multiple low-rank matrices, which reduces FC parameters by more than 80%. We implement extensive experiments on object detection (PASCAL VOC [14]), semantic segmentation (ADE20K [62]), and instance segmentation (MS COCO [28]) to verify the capability of LoRand. Experimental results show that LoRand-Tuning is comparable to fine-tuning on multiple tasks with only 1.8% to 2.8% new backbone parameters, which suggests that the pre-trained backbone parameters can be fully shared. More interestingly, our method completely outperforms fine-tuning on the PASCAL VOC dataset, illustrating that LoRand-Tuning can reduce the impairment of fine-tuning on pre-trained models in low-resource configurations. Our method demonstrates that the LoRand-Tuning paradigm can substantially save storage resources and achieve competitive performances on most dense prediction tasks. In summary, our contributions are three-fold:

- We demonstrate that visual pre-trained models are highly generalisable and shareable. With our training methods, new tasks require only a few trainable parameters to achieve performances comparable to fine-tuning, which can save massive hardware resources.
- We propose the LoRand structure for sparser adapters based on low-rank synthesis. We demonstrate that the backbone parameters in fine-tuning are highly redundant, which can be replaced by 1.8% to 2.8% additional parameters in LoRand.
- Extensive experiments on object detection, semantic segmentation, and instance segmentation show that LoRand-Tuning can achieve remarkable performances and reduce massive new parameters in challenging dense prediction tasks.

2. Related Work

2.1. Training Paradigms in NLP

Computer vision has been continuously inspired by NLP in recent years, including the visual transformer series [5, 13, 29, 32] and self-supervised MAE series [15, 19, 60]. In fact, NLP is leading new training trends different from fine-tuning. Fine-tuning produces a new parameter set for each new task, which is parametrically inefficient for plenty of linguistic tasks [22, 30]. To solve this problem, [30] and [22] have proposed “Prompt Tuning” and “Adapter Tuning” respectively, both of which fix all parameters of the backbone and plug a few tiny trainable structures (less than 10% of the backbone) to adapt the pre-trained model to the new tasks. "Prompt tuning" adds learnable parameters (also known as prompts) to the input or intermediate layers to change the input space of the new tasks. "Prompts" can motivate the model to remember knowledge learned in the previous tasks. "Adapter tuning" adds learnable bottleneck structures after each block to connect the pre-trained model with new tasks. Adapter and prompt demonstrate the coexistence of parameter efficiency and high performances in NLP, stimulating studies in CV. [24] proposes Visual Prompt Tuning (VPT) for image classification and semantic segmentation, but the performance of VPT on semantic segmentation is still far from fine-tuning. This phenomenon motivates us to explore whether adapter tuning can bring a new paradigm in computer vision with fewer parameters and better performances. In this work, we try to explore parameter-efficient and high-performance adapter structures.

2.2. Adapter Tuning

Adapters have been widely studied in NLP. Houlsby *et al.* [22] first add a bottleneck adapter structure to the transformer blocks and fix the original backbone, which achieves comparable performances to fine-tuning. Figure 3 illustrates the differences between fine-tuning and adapter-tuning. [37, 44, 59] further reduce parameters in the adapter with closer performances to fine-tuning. [18, 34, 39] outperform fine-tuning on low-resource tasks, demonstrating that more parameters may not improve performance when fine-tuning pre-trained models [36]. In computer vision, [41] add convolutional adapters to the ResNet [20] and obtain competitive results in image classification. Adapter concept has also been applied in multimodal [33], vision-and-language [51], and domain adaptation [56], but these methods are only applicable under specific conditions. [7, 21, 25, 31] investigate the potential of adapter-tuning for visual classification. [8] apply the adapter structure to visual dense predictions without fixing any original parameters, which indeed trades more parameters for better performances.

2.3. Low-rank Approximation

The low-rank approximation uses multiple low-dimensional tensors to approximate a larger tensor with higher dimensions. Tensor dimensions and sizes in machine learning are very large, so low-rank approximations are widely used in face recognition [61], distributed training [54], transfer learning [11], and cross-domain [10]. A $b \times c$ matrix M can be approximated with N low-rank matrices Q by the following equation:

$$M_{b\times c} = \prod_{i=1}^{N} Q_{r_i \times s_i}, \tag{1}$$

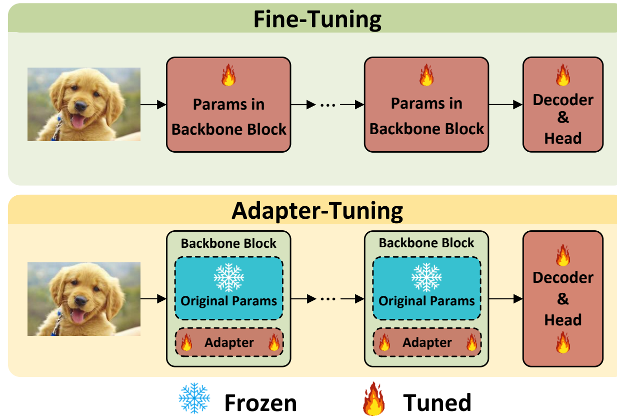


Figure 3. Comparision between Adapter-Tuning and Fine-Tuning paradigms. Fine-Tuning tunes (🔥) all parameters delivered by the pre-trained model. Adapter-Tuning freezes (❄) all structures and parameters in the pre-trained model and only trains (🔥) the additional parameters in adapters. Parameters in the decoder and head are trainable in both paradigms.

where N has different values depending on the approximation methods, we implement low-rank approximation of the adapter matrices by heuristic learning.

3. Method

In this section, we will elaborate on the proposed low-rank adapter (LoRand) in three parts: adapter tuning paradigm, LoRand, and parameter analysis.

3.1. Adapter Tuning Paradigm

For dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, fine-tuning calculates the loss between inference results and labels according to the formula:

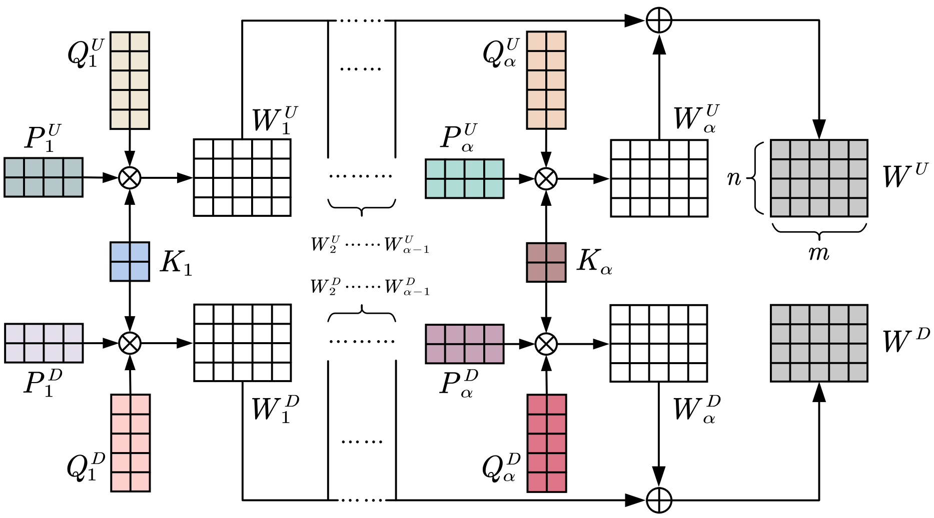
$$L\left(D, \theta\right) = \sum_{i=1}^{N} loss(f_{\theta}\left(x_i\right), y_i), \tag{2}$$

where f_θ denotes the network forward function and $loss$ represents the loss function. After that, θ is optimized through

$$\theta \leftarrow \arg\min_{\theta} L(D, \theta). \tag{3}$$

In adapter tuning paradigm, parameters consist of two parts, including parameters in adapter θ_A and parameters in the original architecture θ. Here, θ is further divided into frozen part θ_F and trainable part θ_T, noted as $\theta = \{\theta_F, \theta_T\}$. Let Ω be all the trainable parameters, then $\Omega = \{\theta_A, \theta_T\}$. The loss function and optimization formula in adapter can be written as:

$$L\left(D, \theta_F, \Omega\right) = \sum_{i=1}^{N} loss(f_{\theta_F, \Omega}\left(x_i\right), y_i), \tag{4}$$



(m,n)	P_{LoRand}	$P_{Adapter}$	%
$(96, 48)$	4736	9216	**51.39%**
$(192, 96)$	9344	36864	**25.35%**
$(384, 192)$	18560	147456	**12.59%**
$(768, 384)$	36992	589824	**6.27%**
······	······	······	······

Figure 4. **Left:** Multi-branch projection in LoRand. The down-projection W^D and up-projection W^U matrices are the summation of α branches $W_1^D(W_1^U)...W_\alpha^D(W_\alpha^U)$. K_i in i-th branch is shared between W_i^D and W_i^U. All the P, Q, and K are trainable, while all the W matrices are calculated. **Right:** Comparisons of the same-sized projection matrices between LoRand and Adapter. (m, n) in the table are typical values in SwinBlocks. LoRand has far fewer parameters than Adapter. With the same projection dimension, LoRand saves over 80% parameters of the Adapter in Swin Transformers. (α, β) here are (2,8), the same as the experiments.

$$\Omega \leftarrow \arg\min_{\Omega} L(D, \theta_F, \Omega). \tag{5}$$

3.2. LoRand

Before introducing LoRand, we first review the existing adapter structure. Conventional adapters are bottleneck structures containing a down-projection, an up-projection, and a non-linear activation function. Besides, adapters ensure the robustness of the model by adding residual [20] structures. Adapter layer can be formulated as follows:

$$A^l = U^l \left(GeLU \left(D^l \left(x \right) \right) \right) + x, \tag{6}$$

where U^l and D^l represent the up and down projections in the l-th adapter layer, and GeLU is the activation function. It is clear that the parameters in adapter come from the projections. The projection process can be written as:

$$y = Wx + b, \tag{7}$$

which means most adapter parameters are in W.

To reduce the adapter parameters, we propose a low-rank adapter (LoRand) structure to replace the W in the projection structures. Figure 2 shows the simplified structure of LoRand. Here we approximate not a specific matrix W but an ideal matrix W_{best} that can transform the feature space of the pre-trained model into new tasks by heuristic learning. The approximation matrix $\hat{W}$ has the same size as W, but the low-rank design makes $\hat{W}$ have far fewer free degrees than a common W.

Specifically, we synthesize each W by multiplying three low-rank matrices $P \in \mathbb{R}^{\beta \times m}, K \in \mathbb{R}^{\beta \times \beta}$, $Q \in \mathbb{R}^{\beta \times n}$, that is:

$$W = P^T K Q, \tag{8}$$

where $\beta \ll \min(m, n)$ ensuring that P and Q are low-rank matrices. K can be regarded as a kernel matrix that controls the parameter size of LoRand.

After that, we add multi-branch structures to LoRand to increase the robustness and stability of low-rank matrices, which is inspired by MoE [43] and adaboost [45,52]. Every W consists of α branches, that is:

$$W = \sum_{i=1}^{\alpha} W_i = \sum_{i=1}^{\alpha} P_i^T K_i Q_i. \tag{9}$$

In addition, we share the kernel matrix K of the two projection layers within each branch. We hope the sharing mechanism can promote the coherence of two projection layers during training process. Besides, the shared K also slightly reduces the number of LoRand parameters. Up to now, the W^U and W^D in a complete LoRand structure can be represented as:

$$W^U = \sum_{i=1}^{\alpha} W_i^U = \sum_{i=1}^{\alpha} (P_i^U)^T K_i Q_i^U, \tag{10}$$

$$W^D = \sum_{i=1}^{\alpha} W_i^D = \sum_{i=1}^{\alpha} (P_i^D)^T K_i Q_i^D, \tag{11}$$

where K_i is shared in W^U and W^D. Figure 4 presents the detailed designs of the multi-branch projection.

3.3. Parameter Analysis

In this section, we will compare the parameters of LoRand and typical adapter [22] with the same size of projection matrix.

Adapter Let m be the input dimension of the adapter and n be the middle layer dimension after down projection. Then the number of parameters in each adapter is $2mn$ (ignoring the few biases). In general, adapter tuning places two adapter modules in each block, so the space complexity of all adapter parameters in γ blocks can be written as:

$$O(4\gamma mn). \tag{12}$$

LoRand According to section 3.2, each W contains α sets of $\{P, Q, K\}$, that is:

$$\alpha(m\beta + \beta^2 + n\beta). \tag{13}$$

Each LoRand consists of two W and α shared K, so the parameter quantity of each LoRand is:

$$2\alpha\left(m\beta + \beta^2 + n\beta\right) - \alpha\beta^2 = 2\alpha\beta(m + n + \beta/2). \tag{14}$$

Each block has two LoRand structures, so the number of parameters in γ blocks is:

$$4\alpha\beta\gamma\,(m + n) + 2\alpha\beta^2\gamma. \tag{15}$$

As $\alpha, \beta, \gamma \ll \min(m, n)$, the space complexity here can be written as:

$$O\left(4\alpha\beta\gamma\,(m + n)\right). \tag{16}$$

Comparison between Formulas 12 and 16 can be simplified as:

$$O(mn), \tag{17}$$

and

$$O(\alpha\beta(m + n)). \tag{18}$$

Given that $\alpha, \beta \ll \min(m, n)$, the space complexity of LoRand is far lower than the typical adapter. The table in Figure 4 illustrates that LoRand saves most Adapter parameters with the same projecting dimension.

4. Experiments

We evaluate LoRand on multiple dense prediction tasks, including object detection, semantic segmentation, and instance segmentation. We also evaluate LoRand under low-resource conditions. We first describe our experimental setup in Section 4.1, including pre-trained backbones, baselines, LoRand settings, and downstream tasks. Then we present the main results of three benchmarks in Section 4.2. We also implement ablation study in Section 4.3 to investigate the impact of structural settings in LoRand.

4.1. Experimental Setup

Pretrained Backbones We conduct experiments on the advanced Swin Transformer [32] architectures. All backbones in this section are pre-trained by ImageNet-22k [9]. Pre-trained models are provided by OpenMMLab [6].

Baselines We compare LoRand with three other common training methods:

(a) FULL: update all parameters in the architecture.

(b) FIXED: fix pre-trained parameters in Swin and train other parts of the architecture (neck, head).

(c) ADAPTER: add two trainable adapter structures in each SwinBlock following [22], and freeze other parts of the backbone. We evaluate two forms of adapter with different middle layer dimensions (D_{ML}):

- ADAPTER-B: D_{ML} is a half of input dimension.
- ADAPTER-T: D_{ML} is a quarter of input dimension.

LoRand Settings We conducted experiments on three LoRand variants, which have different branch numbers α and kernel matrix dimensions β.

- LoRand: $\alpha = 2$, $\beta = 8$ (Standard).
- LoRand+: $\alpha = 4$, $\beta = 8$.
- LoRand++: $\alpha = 4$, $\beta = 16$.

Downstream Tasks We conducted experiments on COCO [28], ADE20K [62], and PASCAL VOC [14] benchmarks to widely evaluate LoRand's performance on main dense prediction tasks.

COCO 2017 [28] is the most commonly used dataset for object detection and instance segmentation, which contains 118K training and 5K validation images. We perform experiments on the validation set. For a fair comparison, all experiments performed on COCO employ Cascade MASK R-CNN [32] as the detector.

ADE20K [62] is the most widely used semantic segmentation dataset, which contains 20K training and 2K validation images. We also conduct experiments on the ADE20K validation set and utilise UperNet [57] as the framework.

PASCAL VOC 0712 [14] is also widely used in object detection, which contains about 16K training and 5K validation images. VOC 0712 is much smaller than the latest benchmarks, so we treat it as a low-resource case. We adopt Faster RCNN [42] as the detector for VOC 0712.

All our experiments are conducted with 8x NVIDIA Tesla V100 GPUs. The experiments on PASCAL VOC and

Swin-L (198M)	Trained* Params	%	Δ_{Full}	Extra Structure	Pascal VOC (Faster RCNN)		ADE20K (UperNet)	
					$\mathrm{AP}_{\mathrm{Box}}$	Δ_{LoRand}	mIoU	Δ_{LoRand}
Baselines								
FULL	198.58 M	100.00 %	-	✗	84.43 %	- 2.69 %	53.25 %	+ 1.34 %
FIXED	0.00 M	0.00 %	- 100.00 %	✗	**85.19 %**	**- 1.93 %**	32.21 %	- 19.70 %
ADAPTER-B	32.04 M	16.13 %	- 83.87 %	✓	80.93 %	- 6.19 %	46.23 %	- 5.68 %
ADAPTER-T	16.04 M	8.08 %	- 91.92 %	✓	78.10 %	- 9.02 %	43.51 %	- 8.40 %
Our Methods								
LORAND	**3.59 M**	**1.84** %	**- 98.16 %**	✓	87.12 %	-	50.67 %	-
LORAND+	7.19 M	3.62 %	- 96.38 %	✓	87.63 %	+ 0.51 %	51.13 %	+ 0.46 %
LORAND++	14.24 M	7.17 %	- 92.83 %	✓	**88.11 %**	**+ 0.99 %**	**51.87 %**	**+ 1.20 %**

Table 1. Results of baselines and our methods on Pascal VOC and ADE20K benchmarks. Swin-L is employed as the pre-trained model here. We present the numbers and percentages of trainable backbone parameters on the left and all the performences on the right. $*$ denotes the trainable parameters in backbones.

Swin-B (89M)	Trained* Params	%	Δ_{Full}	Extra Structure	COCO (Cascade Mask R-CNN)			
					$\mathrm{AP}_{\mathrm{Box}}$	Δ_{LoRand}	$\mathrm{AP}_{\mathrm{Mask}}$	Δ_{LoRand}
Baselines								
FULL	89.14 M	100.00 %	-	✗	51.90 %	+ 0.80 %	45.00 %	+ 0.90 %
FIXED	0.00 M	0.00 %	- 100.00 %	✗	15.30 %	- 35.80 %	10.80 %	- 33.8 %
ADAPTER-B	14.38 M	16.13 %	- 83.87 %	✓	46.50 %	- 4.60 %	40.20 %	- 3.90 %
ADAPTER-T	7.20 M	8.08 %	- 91.92 %	✓	43.20 %	- 7.90 %	38.70 %	- 5.40 %
Our Methods								
LORAND	**2.39 M**	**2.76** %	**- 97.24 %**	✓	51.10 %	-	44.10 %	-
LORAND+	4.73 M	5.31 %	- 94.69 %	✓	51.20 %	+ 0.10 %	44.30 %	+ 0.20 %
LORAND++	9.32 M	10.46 %	- 89.54 %	✓	**51.50 %**	**+ 0.40 %**	**44.40 %**	**+ 0.30 %**

Table 2. Results of baselines and our methods on COCO benchmarks. Swin-B is employed as the pre-trained model here. We present the numbers and percentages of trainable backbone parameters on the left and all the performences on the right. $*$ denotes the trainable parameters in backbones.

ADE20K are based on Swin-S, Swin-B, and Swin-L pre-trained models. Limited by GPU memory, the COCO experiments are based on Swin-T, Swin-S, and Swin-B.

4.2. Main Results

We first compare the trainable backbone parameters and performance of these methods on three benchmarks in Tables 1 and 2. Table 1 shows the results of PASCAL VOC and ADE20K datasets based on Swin-L, and Table 2 shows the results of COCO based on Swin-B. From Tables 1 and 2, we can see that:

1) LoRand can effectively address the dilemma of fine-tuning in low-resource situations. Table 1 shows that FIXED outperforms FULL on the PASCAL VOC dataset, which implies that the powerful generalization ability of pre-trained model is severely weakened during fine-tuning. Fine-tuning with low-resource data reduces the feature understanding of pre-trained models, which leads to the poor performance on downstream tasks. LoRand avoids this disadvantage by fixing the original parameters. More importantly, LoRand can absorb features from the new data by its smaller trainable structures. Table 1 indicates that LoRand outperforms FULL and FIXED by 2.69% and 1.93% on the low-resource dataset with only 1.84% trainable backbone parameters. LoRand+ and LoRand++ also outperform FULL by 3.2% and 3.68% with 3.62% and 7.17% backbone parameters. In fact, there are many other common computer vision datasets with similar volumes to the PASCAL VOC, including CUB-200-2011 [55], Oxford 102 Flowers [35], Stanford Cars [27], and Caltech-256 [16]. The prevalence of "Pretrained & Finetuning" leads us to focus more on giant benchmarks, but Table 1 suggests we need a better training paradigm to cope with many low-resource situations in industrial applications. LoRand-Tuning proves to be a competitive candidate who brings promising performance and parameter-efficient approaches to low-resource cases.

2) LoRand effectively balances the number of trainable backbone parameters and downstream task per-

formance. Tables 1 and 2 demonstrate that LoRand (standard) performs very closely to FULL on large benchmarks with only 1.84% to 2.76% trainable parameters. By tuning less than 3.6M backbone parameters, LoRand (standard) achieves 50.67% (mIOU) on ADE20K, and 51.10% (AP_{Box}) / 44.10% (AP_{Mask}) on COCO, which is only about 1.5% off on average compared to FULL. LoRand+ and LoRand++ further reduce the gap between these two paradigms to approximately 1% with slight parameter increases. For Swin-L, LoRand saves about 195M parameters per copy compared to FULL. For Swin-B, LoRand saves about 86 M. These results are interesting, which means we do not have to spend plenty of hardware resources to store these redundant parameters. Industrial service providers deliver thousands of model training tasks every day. With LoRand-Tuning, millions of gigabytes per year for model storage could be saved.

3) LoRand effectively broadens the potential of conventional parameter-efficient adapter structures in dense predictions. From the results, we can draw similar conclusions to [24] that the standard adapter [22] performs worse than fine-tuning on dense predictions. Tables 1 and 2 illustrate that the ADAPTER's performance is far from FULL, although it reduces 80% of trainable backbone parameters. Also adding new structures, LoRand achieves comparable performance to FULL by training fewer parameters than the ADAPTER. Overall, Tables 1 and 2 demonstrate the feasibility of parameter-efficient tuning paradigm in visual dense prediction tasks.

Comparisons with other fine-tuned backbone. We then show the comparisons of LoRand with some other remarkable fine-tuned backbones in Table 3. Table 3a shows the results based on UperNet and ADE20K, and 3b shows the results based on Cascade MASK R-CNN and COCO. Table 3 shows that LoRand (based on Swin-Transformer) can outperform most existing fine-tuned backbones with less than 2M parameters. Compared to these backbones, LoRand not only presents more robust and superior results but also saves massive hardware resources in this era of parameter explosion. Specifically, LoRand (Swin-T) exceeds COCO by 1.9% (AP_{Box}) and 1.2% (AP_{Mask}) with 80.12M fewer new backbone parameters than ResNeXt-101-64. Similarly, LoRand (Swin-L) surpasses 5.82% (mIoU) on ADE20K with 40.41M fewer trainable backbone parameters than ResNet-101.

Comparisons on different backbone scales. In addition to Swin-L and Swin-B, we also conduct extensive experiments on Swin-S and Swin-T. We illustrate the performance of baselines and LoRand on multiple backbones. Figure 5 shows the performance of the six methods on different backbone scales, which includes three Swin variants for each benchmark. As FIXED's performance on COCO and ADE20K is too low to display, we only show FIXED's results in the PASCAL VOC. Figure 5 indicates that the performance of most methods improves as the backbone scale gets larger. For the LoRand series, more parameters bring better performance, but it is still challenging to outperform FULL on large datasets. For the ADAPTER, ADAPTER-B performs better than ADAPTER-T, suggesting that adding extra parameters does help improve adapter-tuning performance. Experiments on Swin variants systematically

(a) Comparisons between LoRand-Tuning and Fine-Tuning on COCO.

Backbone	**Trained Params∗**	$\mathbf{AP_{Box}}$	$\mathbf{AP_{Mask}}$
	Fine-Tuning Paradigm		
ResNet-101	44 M	47.9 %	41.5 %
ResNeXt-101-32	40 M	48.1 %	41.6 %
ResNeXt-101-64	81 M	48.3 %	41.7 %
DeiT-S	22 M	48.0 %	41.4 %
Swin-T	29 M	50.5 %	43.7 %
Swin-S	50 M	51.8 %	44.7 %
Swin-B	88 M	51.9 %	45.0 %
	LoRand-Tuning		
LoRand (Swin-T)	0.88 M	50.2 %	42.9 %
LoRand (Swin-S)	1.80 M	50.7 %	43.8 %
LoRand (Swin-B)	2.39 M	51.1 %	44.3 %

(b) Comparisons between LoRand-Tuning and Fine-Tuning on ADE20K.

Backbone	**Trained Params∗**	$\mathbf{AP_{Mask}}$
	Fine-Tuning	
ResNet-18	12 M	39.97 %
ResNet-50	25 M	42.78 %
ResNet-101	44 M	44.85 %
DeiT-S	22 M	44.01 %
Swin-S	50 M	49.30 %
Swin-B	88 M	51.60 %
Swin-L	197 M	53.25 %
	LoRand-Tuning	
LoRand (Swin-S)	1.80 M	47.33 %
LoRand (Swin-B)	2.39 M	49.62 %
LoRand (Swin-L)	3.59 M	50.67 %

Table 3. Comparisons between LoRand-Tuning and Fine-Tuning on ADE20K and COCO. We fine-tune multiple backbones and compare their performances with LoRand series. Architectures in (a) and (b) are Cascade Mask R-CNN and UperNet. Parameters in decoder and head are updated in both paradigms. ∗ denotes the trainable parameters in backbones.

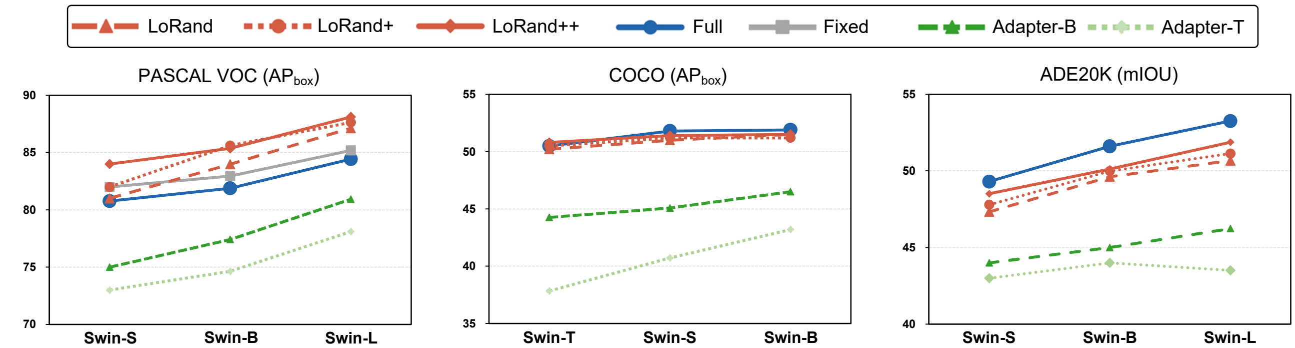


Figure 5. Seven methods on different backbone scales. Figures show results on PASCAL VOC, COCO, and ADE20K from left to right. Swin-S, Swin-B, and Swin-L are employed as the pre-trained models for PASCAL VOC and ADE20K. Swin-T, Swin-S, and Swin-B are employed for COCO. FIXED's performances are so low on COCO and ADE20K that they reduce the intuitiveness of the other six methods, so FIXED is only presented in PASCAL VOC comparisons.

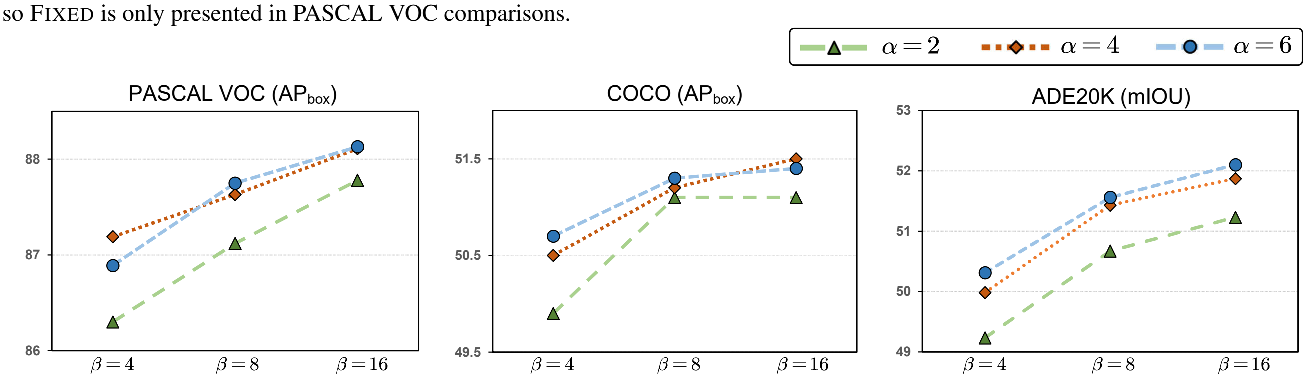


Figure 6. Ablation Study for α and β. α ranges from 2, 4, 6, and β ranges from 4, 8, 16. Figures from left to right present experiments on three benchmarks respectively. We only present $\mathrm{AP_{Box}}$ changes for COCO benchmark considering the strong correlation between the values of $\mathrm{AP_{Box}}$ and $\mathrm{AP_{Mask}}$ in COCO.

demonstrate that LoRand can outperform both FULL and traditional adapter structures in low-resource cases and perform very closely to FULL in large benchmarks.

4.3. Ablation Study

In this section, we ablate two key hyperparameters in LoRand: the LoRand branch number α and the kernel matrix dimension β. α affects the distributed decision-making of LoRand, while β focuses on a single branch's learning capability and consistency.

Several sets of ablation experiments are designed and implemented to investigate the effect of α and β on the performance of LoRand. The ablation experiments were conducted on the same three benchmarks. In order to improve the upper limit of LoRand, our experiments are conducted on the largest backbone of each dataset (ADE20K/PASCAL VOC: Swin-L, COCO: Swin-B). The value sets of α and β are $\{2, 4, 6\}$ and $\{4, 8, 16\}$. Figure 6 shows the results of ablation studies on three datasets. In most cases, LoRand's performance increases slightly as α and β become larger but hardly outperforms fine-tuning on large benchmarks. Besides, exponentially increasing the size of the LoRand does not result in an equivalent performance improvement and even leads to a reduction (α=6 in VOC and COCO). Ablation studies demonstrate that larger LoRands have fewer gains both in parameter efficiency and performance. We have considered this trade-off when designing the LoRand standard, LoRand+, and LoRand++.

5. Conclusion

This paper presents LoRand, a parameter-efficient low-rank adapter for dense predictions, which completely shares the feature understanding of advanced pre-trained models and effectively transfers it to downstream tasks. LoRand performs on par with fine-tuning in COCO instance segmentation, ADE20K semantic segmentation, and PASCAL VOC object detection with only 1% to 3% trainable backbone parameters. Moreover, LoRand effectively avoids the disadvantages of the fine-tuning paradigm and delivers better performance in low-resource situations. We hope that parameter-efficient LoRand can save massive redundant storage resources and facilitate a unified training paradigm for vision and language.

References

[1] Caisse Amisse, Mario Ernesto Jijón-Palma, and Jorge Antonio Silva Centeno. Fine-tuning deep learning models for pedestrian detection. *Boletim de Ciências Geodésicas*, 27, 2021. 1

[2] Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. Cloze-driven pretraining of self-attention networks. *arXiv preprint arXiv:1903.07785*, 2019. 1

[3] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. 1

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 2

[6] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 5

[7] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *arXiv preprint arXiv:2205.13535*, 2022. 3

[8] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022. 3

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1, 2, 5

[10] Zhengming Ding and Yun Fu. Deep transfer low-rank coding for cross-domain learning. *IEEE transactions on neural networks and learning systems*, 30(6):1768–1779, 2018. 3

[11] Zhengming Ding, Ming Shao, and Yun Fu. Deep low-rank coding for transfer learning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015. 3

[12] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020. 2

[13] 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. 2

[14] Mark Everingham, SM Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015. 2, 5

[15] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. *arXiv preprint arXiv:2205.09113*, 2022. 2

[16] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007. 6

[17] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021. 1

[18] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2021. 3

[19] 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. 2

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3, 4

[21] Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. Parameter-efficient fine-tuning for vision transformers. *arXiv preprint arXiv:2203.16329*, 2022. 3

[22] 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, 3, 5, 7

[23] Fatsuma Jauro, Haruna Chiroma, Abdulsalam Y Gital, Mubarak Almutairi, M Abdulhamid Shafi'i, and Jemal H Abawajy. Deep learning architectures in emerging cloud computing architectures: Recent development, challenges and next research trend. *Applied Soft Computing*, 96:106582, 2020. 1

[24] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. *arXiv preprint arXiv:2203.12119*, 2022. 2, 3, 7

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

[26] Christoph Käding, Erik Rodner, Alexander Freytag, and Joachim Denzler. Fine-tuning deep neural networks in continuous learning scenarios. In *Asian Conference on Computer Vision*, pages 588–605. Springer, 2016. 1

[27] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 6

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1, 2, 5

[29] Fanfan Liu, Haoran Wei, Wenzhe Zhao, Guozhen Li, Jingquan Peng, and Zihao Li. Wb-detr: Transformer-based detector without backbone. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2979–2987, 2021. 2

[30] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021. 1, 2

[31] Yen-Cheng Liu, Chih-Yao Ma, Junjiao Tian, Zijian He, and Zsolt Kira. Polyhistor: Parameter-efficient multi-task adaptation for dense vision tasks. *arXiv preprint arXiv:2210.03265*, 2022. 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 the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 1, 2, 5

[33] Cheng Long Li, Andong Lu, Ai Hua Zheng, Zhengzheng Tu, and Jin Tang. Multi-adapter rgbt tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 3

[34] Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. Unipelt: A unified framework for parameter-efficient language model tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264, 2022. 3

[35] 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

[36] Matthew E Peters, Sebastian Ruder, and Noah A Smith. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*, 2019. 2, 3

[37] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *16th Conference of the European Chapter of the Associationfor Computational Linguistics, EACL 2021*, pages 487–503. Association for Computational Linguistics (ACL), 2021. 2, 3

[38] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, 2020. 1, 2

[39] Jonathan Pilault, Christopher Pal, et al. Conditionally adaptive multi-task learning: Improving transfer learning in nlp using fewer parameters & less data. In *International Conference on Learning Representations*, 2020. 3

[40] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020. 1

[41] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017. 3

[42] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 5

[43] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021. 4

[44] Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. Adapterdrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, 2021. 2, 3

[45] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018. 4

[46] Chompunuch Sarasaen, Soumick Chatterjee, Mario Breitkopf, Georg Rose, Andreas Nürnberger, and Oliver Speck. Fine-tuning deep learning model parameters for improved super-resolution of dynamic mri with prior-knowledge. *Artificial Intelligence in Medicine*, 121:102196, 2021. 1

[47] Jaime Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and Pablo Villalobos. Compute trends across three eras of machine learning. *arXiv preprint arXiv:2202.05924*, 2022. 1

[48] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022. 1

[49] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019. 1

[50] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 1

[51] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5227–5237, 2022. 3

[52] B Thilagavathi, K Suthendran, and K Srujanraju. Evaluating the adaboost algorithm for biometric-based face recognition. In *Data Engineering and Communication Technology*, pages 669–678. Springer, 2021. 4

[53] Edna Chebet Too, Li Yujian, Sam Njuki, and Liu Yingchun. A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture*, 161:272–279, 2019. 1

[54] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Practical low-rank communication compression in decentralized deep learning. *Advances in Neural Information Processing Systems*, 33:14171–14181, 2020. 3

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

[56] Xudong Wang, Zhaowei Cai, Dashan Gao, and Nuno Vasconcelos. Towards universal object detection by domain attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7289–7298, 2019. 3

[57] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 418–434, 2018. 5

[58] Sha Yuan, Hanyu Zhao, Shuai Zhao, Jiahong Leng, Yangxiao Liang, Xiaozhi Wang, Jifan Yu, Xin Lv, Zhou Shao, Jiaao He, et al. A roadmap for big model. *arXiv preprint arXiv:2203.14101*, 2022. 1

[59] Aston Zhang, Yi Tay, SHUAI Zhang, Alvin Chan, Anh Tuan Luu, Siu Hui, and Jie Fu. Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters. In *International Conference on Learning Representations*, 2020. 2, 3

[60] Chaoning Zhang, Chenshuang Zhang, Junha Song, John Seon Keun Yi, Kang Zhang, and In So Kweon. A survey on masked autoencoder for self-supervised learning in vision and beyond. *arXiv preprint arXiv:2208.00173*, 2022. 2

[61] Jianwei Zhao, Yongbiao Lv, Zhenghua Zhou, and Feilong Cao. A novel deep learning algorithm for incomplete face recognition: Low-rank-recovery network. *Neural Networks*, 94:115–124, 2017. 3

[62] 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. 2, 5