PRUNING MEETS LOW-RANK PARAMETER-EFFICIENT FINE-TUNING[†]

Mingyang Zhang[‡,†], **Hao Chen**[†], **Chunhua Shen**[†], **Zhen Yang**[†], **Linlin Ou**[‡], **Xinyi Yu**[‡], **Bohan Zhuang**[§]

[†]Zhejiang University, China [‡]Zhejiang University of Technology [§]Monash University, Australia

ABSTRACT

Large pre-trained models (LPMs), such as LLaMA and ViT-G, have shown exceptional performance across various tasks. Although parameter-efficient fine-tuning (PEFT) has emerged to cheaply fine-tune these large models on downstream tasks, their deployment is still hindered by the vast model scale and computational costs. Neural network pruning offers a solution for model compression by removing redundant parameters, but most existing methods rely on computing parameter gradients. However, obtaining the gradients is computationally prohibitive for LPMs, which necessitates the exploration of alternative approaches. To this end, we propose a unified framework for efficient fine-tuning and deployment of LPMs, termed `LoRAPrune`. We first design a PEFT-aware pruning criterion, which utilizes the values and gradients of Low-Rank Adaption (LoRA), rather than the gradients of pre-trained parameters for importance estimation. We then propose an iterative pruning procedure to remove redundant parameters while maximizing the advantages of PEFT. Thus, our `LoRAPrune` delivers an accurate, compact model for efficient inference in a highly cost-effective manner.

Experimental results on various tasks demonstrate that our method achieves state-of-the-art results. For instance, in the VTAB-1k benchmark, `LoRAPrune` utilizes only 0.76% of the trainable parameters and outperforms magnitude and movement pruning methods by a significant margin, achieving a mean Top-1 accuracy that is 5.7% and 4.3% higher, respectively. Moreover, our approach achieves comparable performance to PEFT methods, highlighting its efficacy in delivering high-quality results while benefiting from the advantages of pruning.

1 INTRODUCTION

Large pre-trained models, such as LLaMA [TLI+23] and ViT-G [ZKHB22], have demonstrated exceptional performance in a variety of tasks. However, their remarkable success is accompanied by considerable obstacles stemming from their vast scale and substantial computational costs, thereby making deployment exceedingly arduous [FA23].

Neural network pruning [LKD+17; MTK+17], a popular technique for model compression, can significantly reduce the size and complexity of these large models by removing redundant parameters. Most state-of-the-art methods for evaluating parameter importance require gradients of the parameters. For example, Molchanov [MTK+17; MMT+19] introduced a technique that approximates the loss fluctuation caused by pruning using Taylor expansion, with the first-order term used to assess parameter importance. Similarly, Yu [YHW+22] developed a method based on the gradient's saliency score to evaluate parameter importance and Zhang [ZZL+22a] proposed sensitivity smoothing as an approach to compute parameter importance. Furthermore, the pruning process is often incorporated as part of iterative prune-retrain cycles to restore model accuracy [HDK+18; ZGZ+21]. However, fine-tuning parameters and computing its gradients for LPMs are computationally expensive [FA23]. For instance, fine-tuning GPT-3 requires computing and storing 175B gradients in each iteration, which greatly increases the barrier for pruning. To enable pruning on resource-limited hardware, sparseGPT [BMR+20] circumvents the need to compute all weight gradients at once by employing layer-wise

[†]Work was done when M. Zhang was visiting Zhejiang University. L. Ou is the corresponding author.

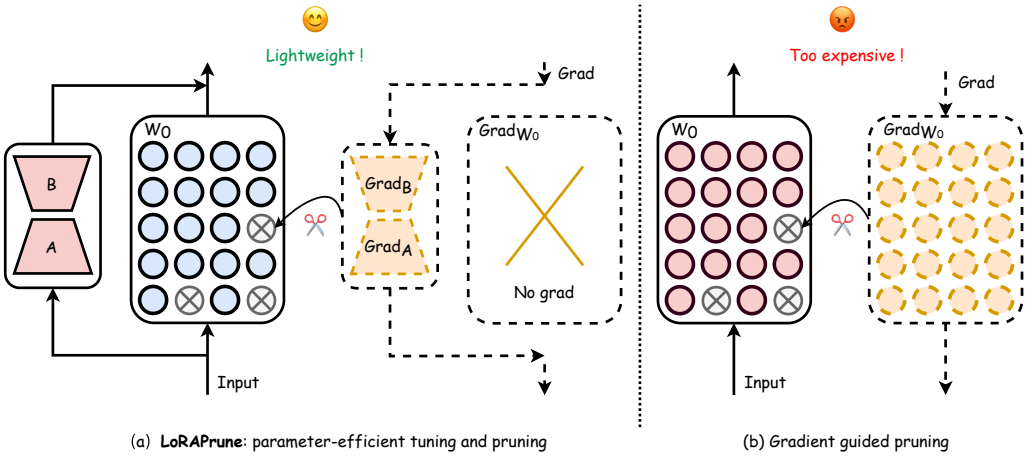

Figure 1: Comparing LoRAPrune (left) with existing gradient-guided pruning approaches (right): (a) LoRAPrune enables efficient tuning and pruning of large-scale models by solely computing low-rank matrices throughout the whole process. (b) Conventional pruning methods necessitate obtaining gradients from substantial-scale parameters. Color(red) indicates trainable parameters, Color (blue) signifies frozen parameters, and Color (yellow) represents gradients.

pruning and reconstruction. However, sparseGPT still has two main drawbacks: **1)** layer-wise pruning is often sub-optimal as a pruning strategy [CDZM20], and **2)** sparseGPT is task-agnostic, which may not yield optimal performance on downstream tasks [HysW+22].

To efficiently fine-tune LPMs, numerous parameter-efficient tuning (PEFT) methods [JTC+22; HysW+22; CGT+22] have been proposed, among which Low-Rank Adaptation (LoRA)-based [HysW+22; LHZ+23; HCZ+23] fine-tuning has gained widespread use. Specifically, LoRA inserts a set of trainable low-rank matrices parallel or serially to the parameters that need to be tuned. The number of parameters in the inserted low-rank matrices is only about 1% of the model's parameters. During fine-tuning on downstream tasks, the original parameters are frozen (i.e., not updated and no gradients are computed), and only the inserted low-rank matrices are updated to approximate the parameter update. Since LoRA updates only a small number of parameters, it exhibits significantly lower optimization difficulty and computational requirements compared to full-parameter fine-tuning methods. However, PEFT typically requires frozen pre-trained parameters without computing their gradients, pruning approaches that rely on gradients of the pre-trained parameters cannot be directly applied to these LPMs.

Thus, a question naturally arises: can the gradients of these low-rank matrices of LoRA be utilized to assess the importance of the pre-trained parameters? In this paper, we propose a unified framework for efficient fine-tuning and deployment, named `LoRAPrune`. To efficiently estimate the importance of pre-trained parameters, `LoRAPrune` uses a novel criterion by only employing the gradients of LoRA. In contrast to the gradient-guided pruning method depicted in Figure 1 (b), `LoRAPrune` leverages LoRA's gradient as an approximation of the pre-trained parameter gradients. consequently `LoRAPrune` accomplishes the objective of pruning the frozen parameters within LPMs. Based on the proposed criterion, we compute the importance of pre-trained parameters in every batch of data and update the importance using a sliding average. After optimizing for a certain number of epochs, the unimportant parts of the pre-trained parameters are gradually pruned. The sliding average and gradual pruning are employed because the model is initially unoptimized for the downstream task, making it challenging to accurately evaluate the importance of each parameter in one shot.

We demonstrate the effectiveness of `LoRAPrune` on several benchmark datasets and show that it achieves state-of-the-art results while significantly reducing the number of parameters and computations. This paper has the following key contributions:

- We introduce a novel parameter importance criterion for large pre-trained models that seamlessly work with LoRA. With the gradients of the low-rank decomposition, we can approximate the importance of pre-trained parameters without needing to compute their gradients.

- Based on the proposed criterion, we introduce `LoRAPrune`, an approach that unifies PEFT with pruning. Compared to other PEFT methods, `LoRAPrune` enables the deployment of lightweight pre-trained models with a similar number of training parameters.
- Experiments on computer vision and natural language processing tasks demonstrate that our `LoRAPrune` outperforms the compared pruning methods and achieves competitive performance with other PEFT methods that do not prune parameters.

2 Related Work

Parameter-efficient tuning. With the prevailing of large pre-trained models, fine-tuning them for downstream tasks has become increasingly costly. Therefore, PEFT methods have received increasing attention from both academia and industry. The mainstream PEFT methods can be divided into adapter-based [HysW+22; LHZ+23; HCZ+23] and prompt-based [JTC+22; WMSDlT22] methods. The former inserts a small fully connected layer near the layer that needs fine-tuning, while the latter adds a few learnable extra tokens to the input sequence of a transformer. However, most PEFT methods require additional parameters to be retained during inference, which can reduce the inference speed. To avoid this, LoRA [HysW+22] proposes inserting two low-rank matrices into the branch of the layer that needs fine-tuning, with no non-linear activation layers in between. Using the re-parameterization principle, the low-rank matrices introduced by LoRA can be fused into the original parameters of the large model after fine-tuning, thereby avoiding additional inference computations. In addition, RepAdapter [LHZ+23] builds on LoRA by proposing the insertion of low-rank matrices in front of the layer that needs fine-tuning, which can also avoid additional inference overhead through re-parameterization.

However, all previous PEFT methods have only been designed for low parameter number during training, and the parameter number remains unchanged during inference. In comparison, the proposed `LoRAPrune` method can ensure low parameter for both tuning and inference.

Neural network pruning. Removing unimportant parameters from large-scale neural networks to reduce the storage and bandwidth requirements and the computational cost of deploying has become a common approach for model compression. However, determining the importance of parameters in a network is still an open question [BGOFG20]. A common approach to model pruning is to use parameter magnitude [LQJ+18; LPM+20; EKT20; HPTD15; LKD+17] as a criterion to determine which weights to prune. This criterion defines the importance of a weight based on its magnitude, such that weights with small magnitudes are pruned. However, this simple criterion does not fully capture a weight's contribution to the model output. This is because small weights can still have a significant impact on the model output due to the complex structure of neural networks, while large weights may not be as important. Many methods use gradient information as part of the criterion for parameter importance, such as optimal brain damage [LDS89; LL16] and its extended criterion [MTK+17; MMT+19; SWR20; YHW+22; ZZL+22a; LAT19; YSRZ22; WZG20], optimal brain surgeon [HSW93; DCP17; ZU19; WGFZ19; SA20] and empirical sensitivity [BLG+18; LBL+19; XS20]. Compared to judging importance based solely on parameter size, gradient-based methods are task-specific and therefore often achieve better performance.

However, the criterion based on gradients cannot be directly applied to PEFT methods since PEFT methods only compute and update the newly inserted parameters while the pre-trained parameters are frozen. PST [LLT+22] proposed using an extra low-rank matrices to learn the gradients of pre-trained parameters. Nevertheless, the gradients are hard to be learned accurately in few-shot setting. In this paper, we propose a method that directly estimates the gradient of the pre-trained parameters using the gradient of LoRA, thus solving the problem of gradient-based methods being difficult to use in large-scale fine-tuning. Compared with PST, our method is more parameter-efficient since we do not need extra low-rank matrices to learn gradients.

3 Method

In this section, we propose `LoRAPrune`, a new approach that unifies PEFT with pruning. We start by reviewing the mainstream LoRA-based PEFT methods. We then propose a novel criterion, which can use LoRA gradients to approximate the importance of pre-trained parameters for pruning. The

overview of LoRAPrune can be found in Figure 1. Compared with traditional gradient-based pruning methods that require computing gradients for all pre-trained parameters, LoRAPrune is lightweight as it only computes the gradients from LoRA.

3.1 PRELIMINARY

We first revisit the parameter-efficient adaption methods with structural re-parameterization. To fine-tune neural network parameter-efficiently, the target modules (e.g., fully connected layer) can be inserted a LoRA to the pre-trained parameter in a parallel or sequential way. During training, the pre-trained parameter is frozen and does not compute its gradient while the inserted LoRA is trainable.

Parallel low-rank adaption. Given two low-rank matrices $A \in \mathbb{R}^{r\times k}$ and $B \in \mathbb{R}^{d\times r}$ ($r \ll \min(d,k)$), the forward process of the target module fine-tuned by parallel low-rank adaption can be written as

$$z = xW_0 + xBA, \tag{1}$$

where W_0, $z \in \mathbb{R}^{n\times k}$ and $x \in \mathbb{R}^{n\times d}$ denote the original target module weights, outputs and inputs of the target module, separately. After adaption, the new weights W can be re-parameterized as $W = W_0 + BA$.

Sequential low-rank adaption. Except for being inserted into an extra branch, the adaptation matrices also can be deployed sequentially. Given two low-rank matrices $A \in \mathbb{R}^{r\times d}$ and $B \in \mathbb{R}^{d\times r}$, the forward process of the target module in sequential low-rank adaption can be written as

$$z = x(BA + E)W_0, \tag{2}$$

where $E \in \mathbb{R}^{d\times d}$ is the identity matrix of size d. After adaption, the new weights can be re-parameterized as $W = (BA + E)W_0$.

3.2 LOW-RANK GRADIENT CRITERION

In a vanilla pruning approach [MTK+17; MMT+19], the importance of a parameter $w_{ij} \in W_0$ can be quantified by the loss induced by removing it. For an input x and corresponding label y, the induced error of w_{ij} can be measured as a squared difference of prediction error with and without the parameter:

$$\mathcal{I}_{ij} = (\mathcal{L}(x, y, W_0) - \mathcal{L}(x, y, W_0|w_{ij} = 0))^2. \tag{3}$$

Computing $\mathcal{I}_{ij}$ for each parameter is computationally expensive. Following [MMT+19], we use first-order Taylor expansion to approximate the importance $\hat{\mathcal{I}}_{ij}$ by:

$$\hat{\mathcal{I}}_{ij} = (\frac{\partial \mathcal{L}}{\partial w_{ij}} w_{ij})^2. \tag{4}$$

However, obtaining the gradient of W_0 in a LPM is difficult since it requires a lot of computing power and storage space. In this work, we discuss how to prune the pre-trained parameter W_0 by inserting the learnable matrices A and B in the downstream task adaption. As mentioned above, the A and B can be inserted into the pre-trained model in a parallel or sequential manner. Therefore, we discuss the corresponding pruning method in these two cases separately.

Pruning for parallel adapter. In the parallel case, we can set the element $(BA)_{ij} = -w_{ij}$ if the element $w_{ij} \in W$ is removed. The importance of each parameter in Eq. (3) can be reformulated as follow

$$\mathcal{I}_{ij} = (\mathcal{L}(x, y, W) - \mathcal{L}(x, y, W|(BA)_{ij} = -w_{ij})^2. \tag{5}$$

Exploiting the first-order Taylor expansion with $(BA)_{ij} = -w_{ij}$ to approximate Eq. (5), the estimated importance $\hat{\mathcal{I}}_{ij}$ of parameter w_{ij} can be represented by

$$\hat{\mathcal{I}}_{ij} = (\frac{\partial \mathcal{L}}{\partial (BA)_{ij}}((BA)_{ij} + w_{ij}))^2. \tag{6}$$

However, preserving the gradient of $\frac{\partial \mathcal{L}}{\partial (BA)_{ij}}$ still entails the same level of complexity as $\frac{\partial \mathcal{L}}{\partial w_{ij}}$, which poses a memory challenge. Here, we only save and use the gradient of two low-rank matrices A and B to approximate the gradient of $\frac{\partial \mathcal{L}}{\partial (BA)}$.

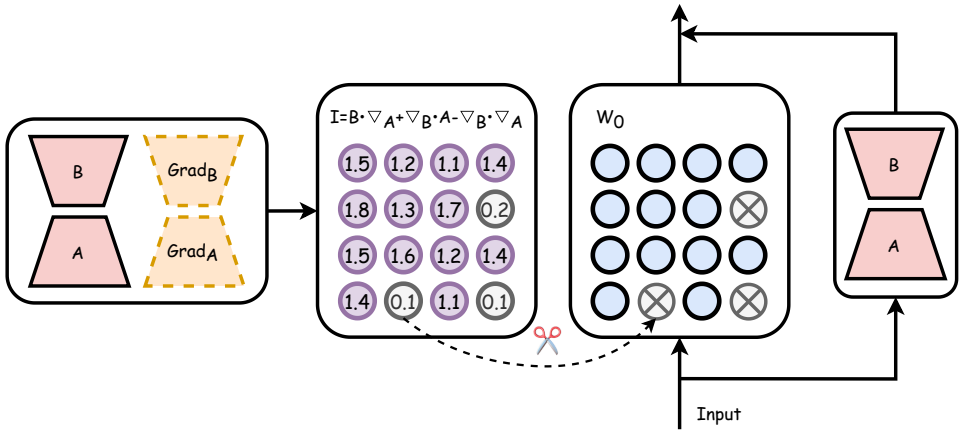

Figure 2: The pruning process for the LoRA gradient criterion involves utilizing the low-rank matrices A, B and their respective gradients ∇_A, ∇_B to compute the importance $\mathcal{I}$. Subsequently, parameters with low scores are removed during the forward pass. The Color (red) represents trainable parameters, while the Color (blue) signifies frozen parameters. The Color (yellow) denotes gradients, and the Color (purple) illustrates the importance of the parameters.

We can rely on the gradient update that $(BA)_{ij}|_{t+1} = (BA)_{ij}|_t - \eta \frac{\partial \mathcal{L}}{\partial (BA)_{ij}}$ to estimate the gradient, where $(BA)_{ij}|_{t+1}$ and $(BA)_{ij}|_t$ represents the $(BA)_{ij}$ in $(t+1)$-th and t-th step, respectively. For simplicity, we ignore the learning rate η since it has no influence on the final result. Apparently, $\frac{\partial \mathcal{L}}{\partial (BA)_{ij}}$ is proportional to the change of BA, which can be written as

$$\frac{\partial \mathcal{L}}{\partial (BA)_{ij}} \propto [(BA)_{ij}|_t - (BA)_{ij}|_{t+1}]. \tag{7}$$

Here, $(BA)_{ij}|_{t+1} = B_{i:}|_{t+1} A_{:j}|_{t+1}$ is generated by the multiplication of i-th row of $B|_{t+1}$ and j-th column of $A|_{t+1}$. Using the above assumption, we can also estimate $\frac{\partial \mathcal{L}}{\partial A_{:j}} \propto A|_{t+1} - A|_t$ and $\frac{\partial \mathcal{L}}{\partial B_{i:}} \propto B|_{t+1} - B|_t$, respectively. Subsequently, we substitute $(BA)_{ij}$ to Eq. (7) and obtain

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial (BA)_{ij}} &\propto [B_{i:}A_{:j} - (B_{i:} - \frac{\partial \mathcal{L}}{\partial B_{i:}})(A_{:j} - \frac{\partial \mathcal{L}}{\partial A_{:j}})], \\ &= [\frac{\partial \mathcal{L}}{\partial B_{i:}} A_{:j} + B_{i:} \frac{\partial \mathcal{L}}{\partial A_{:j}} - \frac{\partial \mathcal{L}}{\partial B_{i:}} \frac{\partial \mathcal{L}}{\partial A_{:j}}]. \end{aligned} \tag{8}$$

where $\frac{\partial \mathcal{L}}{\partial A_{:j}}$ and $\frac{\partial \mathcal{L}}{\partial B_{i:}}$ denote the gradient of A in j-th column and B in i-th row, respectively. Substitute Eq. (8) to Eq. (6), we can estimate the importance of each parameter in a gradient-based manner. With this approximation, we can maintain two low dimensional gradient vectors instead of a large one of size $d \times k$, which leads to our efficient implementation.

Pruning for sequential adapter. As mentioned in Sec. 3.1, the new weight in parameter-efficient tuning can be denoted as $W = (BA + E)W_0$. Therefore, we can readily rewrite the importance function in Eq. (3) by using first-order Taylor expansion

$$\hat{\mathcal{I}}_{ij} = (\frac{\partial \mathcal{L}}{\partial ((BA + E)W_0)_{ij}} ((BA + E)W_0)_{ij})^2. \tag{9}$$

Similar to pruning for a parallel adapter, we also can estimate the gradient of W based on gradient descent as follows

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial ((BA + E)W_0)_{ij}} &\propto [(B_{i:}A_{:j} + E_{ij})w_{ij} - ((B_{i:} - \frac{\partial \mathcal{L}}{\partial B_{i:}})(A_{:j} - \frac{\partial \mathcal{L}}{\partial A_{:j}}) + E_{ij})w_{ij}], \\ &= [(\frac{\partial \mathcal{L}}{\partial B_{i:}} A_{:j} + B_{i:} \frac{\partial \mathcal{L}}{\partial A_{:j}} - \frac{\partial \mathcal{L}}{\partial B_{i:}} \frac{\partial \mathcal{L}}{\partial A_{:j}})w_{ij}]. \end{aligned} \tag{10}$$

As shown in Figure 2, the LoRA gradient criterion only needs to compute the gradient of A and B, which saves memory and computation compared with the gradient of total pre-trained weights W_0.

Algorithm 1: Progressive pruning with LoRA gradient criterion. $|\cdot|$ denotes the total number of elements in the matrix.

Require : Observed data $\mathcal{D}$; Original weights W_0; Randomly initialized low-rank matrices A and B; Loss function $\mathcal{L}$; Sparsity ratio s; Training iterations T.

Output : Trained low-rank adaption A and B; Binary mask $\mathcal{B}$.

$\hat{\mathcal{I}}_{ij}^l \leftarrow 0, \mathcal{B}_{ij}^l \leftarrow 1 \; \forall l, \forall i, \forall j$; `// Initialize Importance Score and Binary mask`

for $t \in [1, \ldots, T]$ **do**

  Clear gradient;

  Forward and backward via Eq. (12);

  Update via the optimizer;

  Calculate $\hat{\mathcal{I}}_t$ via Eq. (9) or (6);

  Calculate $\bar{\mathcal{I}}_t$ via Eq. (11);

  if $\frac{||\mathcal{B}^l||_0}{|\mathcal{B}^l|} \neq s$ **then**

    for $l \in [1, \ldots, L]$ **do**

      $\mathcal{B}_{ij}^l \leftarrow 0$ where $\bar{\mathcal{I}}_{ij}^l$ in Top-k smallest; `// Remove Unimportant Parameters`

    end

  end

end

3.3 LoRA Pruning

Based on the proposed LoRA gradient criterion, we propose `LoRAPrune`, a method that progressively prunes unimportant parameters in large-scale models. The precondition for using Eq. (5) is that the new parameter W has been trained effectively in the downstream task. However, at the beginning of parameter-efficient fine-tuning, the inserted adapters are not optimized well. Although, one option is to tune the new parameters first and then iteratively prune the unimportant parameters, which is time-consuming. To overcome the challenge, we utilize moving average to assess parameter importance. Specifically, the parameter importance at t-th iteration is computed as follows:

$$\bar{\mathcal{I}}_t = \lambda \hat{\mathcal{I}}_{t-1} + (1 - \lambda)\hat{\mathcal{I}}_t. \tag{11}$$

Here, $\bar{\mathcal{I}}_t$ denotes the importance score calculated by Eq. (6) or Eq. (9) at the t-th iteration, and $\lambda \in [0, 1]$ balances the importance between historical and current values.

We first insert a binary mask $\mathcal{B} \in \{0, 1\}^{d \times k}$ for parameters and then use the prune-finetune-prune method for pruning. Given the importance of each parameter, we retain the Top-k important parameters during each pruning iteration by setting the corresponding mask to 1 and setting the rest to 0. Formally, the forward process of each pruned layer can be writen as

$$z = x(W \odot \mathcal{B}), \tag{12}$$

where $\odot$ denotes Hardamard product. Our `LoRAPrune` is outlined in Algorithm 1. In each iteration, `LoRAPrune` performs forward and backward propagation using Eq. (12) and updates the low-rank adaption via the optimizer. Then, the current importance score is computed by using either Eq. (9) or (6). The current importance score is used to calculate the moving average importance score using Eq. (11). If the number of remaining parameters in the binary mask is not equal to the sparsity ratio, the algorithm removes the Top-k unimportant parameters by setting the corresponding elements in the binary mask to zero. It is noted that k will gradually increase with the number of iterations.

4 Experiments

4.1 Experimental Setup

Datasets and metrics. We evaluate our `LoRAPrune` on total 32 downstream tasks, including computer vision (CV) and natural language processing (NLP). 1) The VTB-1k benchmark consists of 19 image classification datasets that span a diverse range of scenarios. These datasets are grouped into three

ViT-B/16 (85.8M)	Sparsity ratio	FGVC		VTAB-1k				
		Tuned / Total	Mean Acc.	Tuned / Total	Natural	Specialized	Structured	Mean Acc.
Full	0%	100.00%	88.5	100.00%	75.9	83.4	47.6	69.0
Pruning methods								
MaP	50%	100.00%	84.6	100.00%	71.3	81.7	45.9	66.3
MvP	50%	100.00%	84.9	100.00%	72.8	80.5	49.8	67.7
RaP	50%	0.75%	73.1	0.75%	53.8	70.4	41.3	55.2
MaP-LoRA	50%	0.75%	83.1	0.75%	75.4	81.9	53.6	70.3
PST	50%	2.14%	85.1	2.14%	73.4	**82.7**	51.3	69.1
LRP-Par (Ours)	50%	0.75%	<u>85.4</u>	0.75%	**76.6**	<u>82.4</u>	**57.1**	**72.0**
LRP-Seq (Ours)	50%	0.75%	**85.7**	0.75%	<u>76.2</u>	81.7	<u>54.9</u>	<u>70.9</u>
Unpruning methods								
Linear	0%	0.12%	79.3	0.04%	68.9	77.2	26.8	57.6
Partial-1	0%	8.38%	82.6	8.30%	69.4	78.5	34.2	60.7
VPT	0%	0.75%	<u>88.4</u>	0.75%	78.5	82.4	55.0	72.0
LoRA-8	0%	0.55%	86.0	0.23%	79.5	84.6	60.5	74.9
LoRA-16	0%	0.90%	84.8	0.69%	<u>79.8</u>	<u>84.9</u>	<u>60.2</u>	<u>75.0</u>
SPT-LoRA	0%	0.41%	**89.3**	0.31%	**81.5**	**85.6**	**60.7**	**75.9**

Table 1: Comparisons on FGVC and VTAB-1k [ZPK+19] benchmarks using ViT-B/16 pre-trained on ImageNet-21k. Sparsity ratio denotes the ratio of pruned parameters , and "Tuned/Total" denotes the fraction of trainable parameters. The best result is in **bold**, and the second-best result is <u>underlined</u>.

categories: *Natural*, *Specialized*, and *Structured*. Each dataset comprises 800 and 200 samples for training and validation, respectively. In accordance with prior work [JTC+22; ZZL22b], we utilize all available samples from the *train* and *val* splits to train models and evaluate their Top-1 accuracy on the *test* split. 2) The FGVC benchmark evaluates models for fine-grained visual classification on five datasets: CUB-200-2011 [WBW+11], NABirds [VHBF+15], Oxford Flowers [NZ08], Stanford Cars [GKW+17], and Stanford Dogs [KJYFF11]. Each dataset contains between 55 to 200 classes and several thousand images for training, validation, and testing. In the absence of a validation set, we use the validation splits in [JTC+22]. 3) The GLUE benchmark is used, which consists of nine natural language understanding (NLU) tasks including natural language inference, text entailment, sentiment analysis, and semantic similarity, among others. The benchmark comprises CoLA, SST-2, MRPC, STS-B, QQP, MNLI, QNLI, RTE.

Implementation details. For CV tasks, we experiment with three different pre-training strategies, namely supervised pre-training, and self-supervised pre-training with MAE [HCX+22] and MoCov3 [CXH21], using the plain vision Transformer backbone ViT-B/16 [DBK+21]. The fine-tuning strategies are adapted from [HCZ+23]. We set the batch size, learning rate, and weight decay as 64, 1×10^{-3}, and 1×10^{-4}, respectively. As for NLP tasks, we employ BERT-base as the pre-trained model. Following [LLT+22], we set the batch size to 32 and perform a hyperparameter search over learning rate $\in$ {3e-5, 5e-5, 1e-4, 5e-4} and epoch $\in$ {20, 40}. Both CV and NLP models are optimized by AdamW optimizer [HDL+20] with cosine learning rate decay. All experiments are conducted on one RTX 3090 GPU.

For all tasks, the first 10% and last 30% of the fine-tuning epochs exclusively involve parameter updating, without any pruning. We adopt a "prune-finetune-prune" approach during pruning, until the desired sparsity level is achieved, using a cubic sparsity scheduler [SWR20]. All fully connected layers in the Attention and Feed Forward Network (FFN) layers are pruned.

Contenders. Due to the lack of pruning works conducted under PEFT settings, we replicate several pruning methods: **1)** Magnitude pruning (MaP) [LQJ+18] computes the importance of parameters based on their magnitude, making it a data-free pruning method. **2)** Magnitude pruning with LoRA (MaP-LoRA) prunes parameters according to its magnitude and fine-tunes by LoRA. **3)** Movement Pruning (MvP) [SWR20] derives importance from first-order information, making it a data-driven pruning method. **4)** Random Pruning (RaP) [LAL+22] randomly selects parameters to prune and fine-tunes by LoRA. Both original MaP and MvP are pruned and tuned on the pre-trained parameters. **5)** Parameter-efficient sparsity (PST) [LLT+22] uses extra low-rank matrices to learn the gradients of pre-trained parameters.

BERT-Base (110.0M)	Sparsity ratio	GLUE									
		Tuned / Total	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Mean Acc.
Full	0%	100.00%	84.7	87.8	91.5	93.0	58.6	88.7	89.5	62.9	82.0
MaP	50%	100.00%	**83.6**	**87.8**	**91.5**	91.0	**60.1**	**89.8**	90.7	67.2	**82.7**
MvP	50%	100.00%	82.3	87.3	90.8	90.8	57.7	89.4	**91.1**	67.2	82.1
PST	50%	2.14%	81.0	85.8	89.8	**91.3**	57.6	84.6	90.7	67.9	81.0
LRP-Par (Ours)	50%	2.14%	82.4	87.2	89.6	90.9	54.1	88.7	89.8	**69.3**	82.2
MaP	90%	100.00%	78.2	83.2	84.1	85.4	27.9	82.3	80.5	50.1	71.4
MvP	90%	100.00%	**80.1**	84.4	**87.2**	87.2	28.6	**84.3**	84.1	57.6	74.2
PST	90%	2.14%	79.6	**86.1**	86.6	89.0	**38.0**	81.3	83.6	**63.2**	**75.9**
LRP-Par (Ours)	90%	2.14%	79.4	86.0	85.3	**89.1**	35.6	83.3	**84.4**	62.8	75.7

Table 2: Comparisons on GLUE benchmark using BERT-Base backbone. Sparsity ratio denotes the ratio of pruned parameters , and "Tuned/Total" denotes the fraction of trainable parameters. The best result is in **bold**, and the second-best result is underlined.

4.2 Main Results

For the proposed LoRAPrune, we use **LRP-Par** and **LRP-Seq** to represent our proposed pruning method with parallel and sequential low-rank adaption, respectively.

Image classification. Firstly, our proposed `LoRAPrune` outperforms other pruning methods on both FGVC and VTAB-1k datasets, as shown in the Table 1. For example, on the 19 tasks of the VTAB-1k dataset, LRP-Par achieved 72% average Top-1 accuracy that was 4.3% higher than MvP, which prunes using the original parameter gradients. This is because MvP requires fine-tuning of the original parameters during pruning, which can lead to overfitting with limited training data. Moreover, compared to MvP, our LRP-Par only requires 0.75% of the total parameters to be computed during pruning and fine-tuning, which is much less than MaP and MvP methods. When compared with other PEFT methods such as PST [LLT+22], MaP-LoRA, and RaP, our LRP-Par achieves a higher average Top-1 accuracy by 2.9%, 1.7%, and 16.8%, respectively, demonstrating the effectiveness of our proposed LoRA gradient criterion. On the FGVC dataset, LRP-Seq achieves the highest results, with average Top-1 accuracies that were 1.1%, 0.8%, and 0.6% higher than MaP, MvP, and PST [LLT+22], respectively. Secondly, compared to fine-tuning methods without pruning, `LoRAPrune` produces competitive results. For instance, on the VTAB-1k dataset, LRP-Par significantly outperforms Linear and Partial-1, and is on par with the VPT. More results under different pre-trained models can be found in **Appendix**.

Natural language understanding. Table 2 demonstrates the effectiveness of our proposed method. Compared with full fine-tuning methods (MaP and MvP), LRP-Par achieves comparable or even superior performance to them while only fine-tuning 2.14% parameters. Compared with PST [LLT+22] that maintains the same number of trainable parameters as LPR-Par, LPR-Par achieves an average score improvement of 1.1% on the GLUE dataset when the sparsity ratio is 50%. These results clearly shows that our method outperforms existing methods in terms of both model compactness and performance.

4.3 Ablation Study

Effect of LoRA gradient criterion. In this paper, we propose to use the gradients of the low-rank matrices in LoRA to evaluate the gradients of the original parameters. What would happen if we directly use the gradients of the original parameters for evaluation? To answer this question, we unlock the original parameters to participate in backpropagation and calculate their gradients. The gradients of the original parameters are then directly used to replace the gradients estimated by the LoRA matrices in Eq. (6) for pruning. It is worth noting that to ensure the fairness of the experiments, we still only update the parameters of the LoRA matrices during pruning and parameter reconstruction.

We conducted experiments on multiple pruning scenarios with different sparsity ratios using three types of datasets: Natural, Specialized, and Structured. The experimental results, as shown in Figure 3, demonstrate that `LoRAPrune` achieves comparable or even superior performance to the original gradient-based methods on EuroSAT (Specialized) and DMLab (Structured) datasets. In the case of CIFAR-100 (Natural) dataset, `LoRAPrune` exhibits competitive performance compared to the original

ViT-B/16 (85.8M)	Sparsity ratio	VTAB-1k					
		Tuned / Total	Natural	Specialized	Structured	Mean Acc.	Total Epochs
LRP-Par	50%	0.75%	76.6	82.4	57.1	72.0	100
Strategy 1	50%	0.75%	76.4	82.1	57.5	72.0	200
Strategy 2	50%	100.00%	63.8	80.3	49.8	64.6	200
Strategy 3	50%	100.00%	51.2	72.7	38.6	54.2	100

Table 3: Comparisons on VTAB-1k [ZPK+19] benchmarks using different pruning strategies. Sparsity ratio denotes the ratio of pruned parameters, "Tuned/Total" denotes the fraction of trainable parameters, and Total Epochs includes fine-tuning and pruning.

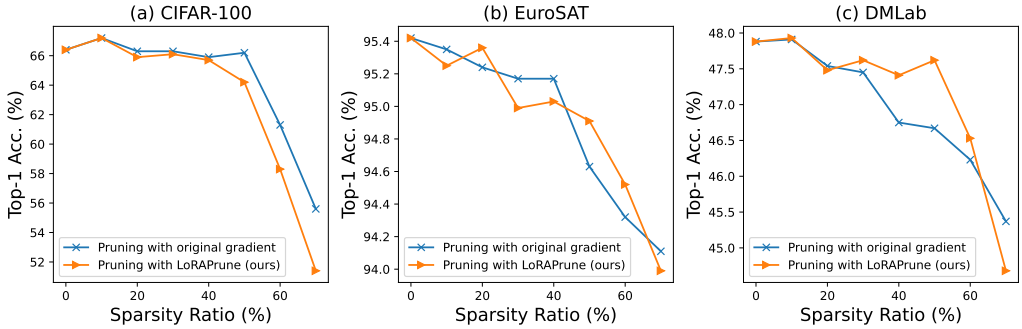

Figure 3: Comparison between LoRAPrune and original gradient-based pruning methods across different sparsity ratios using (a) CIFAR-100, (b) EuroSAT, and (c) DMLab datasets.

gradient-based methods. These findings validate the effectiveness of the criterion proposed in this paper.

Joint *vs.* separate. LoRAPrune performs fine-tuning and pruning simultaneously on downstream tasks. To validate the effectiveness of this approach, We explore the results if we separate the fine-tuning and pruning. Here, we first fine-tune the pretrained-model on downstream tasks and then respectively prune the well-adapted model with the following pruning strategies: **1)** progressively pruning with the proposed criterion by fine-tuning LoRA, **2)** progressively pruning with the original gradient and fine-tuning all parameters, **3)** one-shot pruning and reconstructing the kept parameters with the optimal brain surgeon [HSW93].

The experimental results presented in Table 3 indicate that the proposed method yields comparable performance to Strategy 1 while requiring only half of the training epochs. Moreover, the proposed method outperforms Strategy 2 by 7.4% in terms of the mean Top1 accuracy, despite using only half of the training time and 0.75% of the training parameters. It is worth noting that the pruned model obtained by Strategy 3 performs the worst, demonstrating the effectiveness of progressive pruning. These results show that combining the process of fine-tuning and pruning is time-efficient and does no harm to the performance of PLMs.

5 Conclusion and Future Work

In this paper, we have proposed a method to effectively prune and fine-tune LPMs simultaneously, achieving state-of-the-art efficiency-accuracy trade-offs. Specifically, we propose a novel criterion for evaluating the parameter importance by only computing the low-rank matrices in LoRA, which greatly reduces the computational resources required for pruning large models. Building upon the proposed criterion, we present LoRAPrune, a technique that performs pruning and fine-tuning without the need for computing gradients of the pre-trained weights. Finally, comprehensive experiments on computer vision and natural language processing tasks demonstrate the superiority of LoRAPrune over other pruning methods that necessitate computing gradients for all parameters. With ablation study, we show that using LoRA gradients to evaluate the importance of parameters is efficient and accurate by comparing the effects of using original gradients. In future work, we plan to apply LoRAPrune to even larger models with more parameters, such as ViT-G [ZKHB22], LLaMa-7B [TLI+23].

Limitations and societal impact. Although LoRAPrune demands substantial GPU memory in its pruning process due to low-rank matrix integration, its potential societal benefits are noteworthy. It can decrease energy consumption and carbon emissions by enhancing large-scale model efficiency for training and inference. However, it also increases vulnerability to modifications by malicious actors. Additionally, our method has not yet achieved hardware-software codesign and is currently only effective at the algorithmic level.

Acknowledgements: This work was supported by National Key R&D Program of China (No. 2022ZD0118700).

REFERENCES

[BGOFG20] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proc. Int. Conf. Mach. Learn. and Syst.*, 2:129–146, 2020.

[BLG+18] Cenk Baykal, Lucas Liebenwein, Igor Gilitschenski, Dan Feldman, and Daniela Rus. Data-dependent coresets for compressing neural networks with applications to generalization bounds. *arXiv preprint arXiv:1804.05345*, 2018.

[BMR+20] 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. *Proc. Adv. Neural Inf. Process. Syst.*, 33:1877–1901, 2020.

[CDZM20] Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. Towards efficient model compression via learned global ranking. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1518–1528, 2020.

[CGT+22] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Proc. Adv. Neural Inf. Process. Syst.*, 2022.

[CXH21] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 9640–9649, 2021.

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. Int. Conf. Learn. Repren.*, 2021.

[DCP17] Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *Proc. Adv. Neural Inf. Process. Syst.*, 30, 2017.

[EKT20] Bryn Elesedy, Varun Kanade, and Yee Whye Teh. Lottery tickets in linear models: An analysis of iterative magnitude pruning. *arXiv preprint arXiv:2007.08243*, 2020.

[FA23] Elias Frantar and Dan Alistarh. Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*, 2023.

[GKW+17] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. Fine-grained car detection for visual census estimation. In *Proc. AAAI Conf. on Arti. Intel.*, 2017.

[HCX+22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 16000–16009, 2022.

[HCZ+23] Haoyu He, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. Sensitivity-aware visual parameter-efficient tuning. *arXiv preprint arXiv:2303.08566*, 2023.

[HDK+18] Yang He, Xuanyi Dong, Guoliang Kang, Yanwei Fu, and Yi Yang. Progressive deep neural networks acceleration via soft filter pruning. *arXiv preprint arXiv:1808.07471*, 1(2):8, 2018.

[HDL+20] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2009–2018, 2020.

[HPTD15] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Proc. Adv. Neural Inf. Process. Syst.*, 28, 2015.

[HSW93] Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *Proc. IEEE Conf. on Neural Networks*, pages 293–299, 1993.

[HysW+22] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *Proc. Int. Conf. Learn. Repren.*, 2022.

[JTC+22] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *Proc. Eur. Conf. Comp. Vis.*, 2022.

[KJYFF11] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn. Workshops*, 2011.

[LAL+22] Yawei Li, Kamil Adamczewski, Wen Li, Shuhang Gu, Radu Timofte, and Luc Van Gool. Revisiting random channel pruning for neural network compression. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 191–201, 2022.

[LAT19] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *Proc. Int. Conf. Learn. Repren.*, 2019.

[LBL+19] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. *arXiv preprint arXiv:1911.07412*, 2019.

[LDS89] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Proc. Adv. Neural Inf. Process. Syst.*, 2, 1989.

[LHZ+23] Gen Luo, Minglang Huang, Yiyi Zhou, Xiaoshuai Sun, Guannan Jiang, Zhiyu Wang, and Rongrong Ji. Towards efficient visual adaption via structural re-parameterization. *arXiv preprint arXiv:2302.08106*, 2023.

[LKD+17] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *Proc. Int. Conf. Learn. Repren.*, 2017.

[LL16] Vadim Lebedev and Victor Lempitsky. Fast convnets using group-wise brain damage. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2554–2564, 2016.

[LLT+22] Yuchao Li, Fuli Luo, Chuanqi Tan, Mengdi Wang, Songfang Huang, Shen Li, and Junjie Bai. Parameter-efficient sparsity for large language models fine-tuning. *arXiv preprint arXiv:2205.11005*, 2022.

[LPM+20] Jaeho Lee, Sejun Park, Sangwoo Mo, Sungsoo Ahn, and Jinwoo Shin. Layer-adaptive sparsity for the magnitude-based pruning. *arXiv preprint arXiv:2010.07611*, 2020.

[LQJ+18] Guiying Li, Chao Qian, Chunhui Jiang, Xiaofen Lu, and Ke Tang. Optimization based layer-wise magnitude-based pruning for dnn compression. In *Int. Joi. Conf. on Artificial Intelligence*, pages 2383–2389, 2018.

[MMT+19] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 11264–11272, 2019.

[MTK+17] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *Proc. Int. Conf. Learn. Repren.*, 2017.

[NZ08] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Proc. Conf. on Comp. Vis., Graph. & Image Process.*, pages 722–729, 2008.

[SA20] Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Proc. Adv. Neural Inf. Process. Syst.*, 33:18098–18109, 2020.

[SWR20] Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Proc. Adv. Neural Inf. Process. Syst.*, 33:20378–20389, 2020.

[TLI+23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[VHBF+15] 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 *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 595–604, 2015.

[WBW+11] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *Tech. Rep. CNS-TR-2011-001, California Institute of Technology*, 2011.

[WGFZ19] Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *Proc. Int. Conf. Mach. Learn.*, pages 6566–6575, 2019.

[WMSDlT22] Chen Henry Wu, Saman Motamed, Shaunak Srivastava, and Fernando D De la Torre. Generative visual prompt: Unifying distributional control of pre-trained generative models. *Proc. Adv. Neural Inf. Process. Syst.*, 35:22422–22437, 2022.

[WZG20] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

[XS20] Xin Xing and Long Sha. Probabilistic connection importance inference and lossless compression of deep neural networks. In *Proc. Int. Conf. Learn. Repren.*, 2020.

[YHW+22] Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *Proc. AAAI Conf. on Arti. Intel.*, volume 36, pages 3143–3151, 2022.

[YSRZ22] Xin Yu, Thiago Serra, Srikumar Ramalingam, and Shandian Zhe. The combinatorial brain surgeon: Pruning weights that cancel one another in neural networks. In *Proc. Int. Conf. Mach. Learn.*, pages 25668–25683, 2022.

[ZGZ+21] Chen Zhao, Yixiao Ge, Feng Zhu, Rui Zhao, Hongsheng Li, and Mathieu Salzmann. Progressive correspondence pruning by consensus learning. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 6464–6473, 2021.

[ZKHB22] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 12104–12113, 2022.

[ZPK+19] 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.

[ZU19] Wenyuan Zeng and Raquel Urtasun. Mlprune: Multi-layer pruning for automated neural network compression. 2019.

[ZZL+22a] Qingru Zhang, Simiao Zuo, Chen Liang, Alexander Bukharin, Pengcheng He, Weizhu Chen, and Tuo Zhao. Platon: Pruning large transformer models with upper confidence bound of weight importance. In *Proc. Int. Conf. Mach. Learn.*, pages 26809–26823, 2022.

[ZZL22b] Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. Neural prompt search. *arXiv preprint arXiv:2206.04673*, 2022.

APPENDIX

APPLYING LORAPRUNE TO MORE PRE-TRAINING METHODS

In addition to the backbones pre-trained with ImageNet-21k, we experiment with two self-supervised methods: MAE [HCX+22] and MoCo v3 [CXH21]. The results are shown in Table 4 and we observe that under the self-supervised pre-trained models, LoRAPrune exhibited remarkably impressive pruning results, even outperforming many fine-tuning methods without pruning. For instance, when using MAE-pretrained weights, LPR-Par achieved a higher average Top-1 accuracy than LoRA-8 and LoRA-16 by 1.9% and 1.4%, respectively. This could be attributed to the fact that the pre-training data volume of MAE and MoCo v3 (trained on the ImageNet-1k dataset) was insufficient to fully capture the model's representational capacity. As a result, the model's parameters contained a significant amount of redundancy, which could be pruned more effectively.

ViT-B/16 (85.8M)	Sparsity ratio	VTAB-1k MAE					VTAB-1k MoCo v3				
		Tuned / Total	Natural	Specialized	Structured	Mean Acc.	Tuned / Total	Natural	Specialized	Structured	Mean Acc.
Full	0%	100%	59.3	79.7	53.8	64.3	100%	72.0	84.7	42.0	69.6
LPR-par (Ours)	50%	0.75%	61.3	78.8	58.4	66.2	0.75%	71.5	82.5	58.6	70.9
Linear	0%	0.04%	18.9	52.7	23.7	32.1	0.04%	67.5	81.1	30.3	59.6
Partial-1	0%	8.30%	58.4	78.3	47.6	61.5	8.30%	72.3	84.6	47.9	68.3
Bias	0%	0.13%	54.6	75.7	47.7	59.3	0.13%	72.9	81.1	53.4	69.2
LoRA-8	0%	0.23%	57.5	77.7	57.7	64.3	0.23%	21.2	66.7	45.1	44.3
LoRA-16	0%	0.69%	57.3	77.1	59.9	64.8	0.69%	16.0	64.0	48.7	42.9
SPT-LoRA	0%	0.69%	**65.4**	**82.4**	**61.5**	**69.8**	0.50%	**76.5**	**86.0**	**63.6**	**75.3**

Table 4: Comparisons on VTAB-1k [ZPK+19] benchmark using self-supervised ViT-B/16 backbone pre-trained by MAE [HCX+22] and MoCo v3 [CXH21]. The sparsity ratio denotes the ratio of pruned parameters, and "Tuned/Total" denotes the fraction of trainable parameters. The best result is in **bold**, and the second-best result is underlined.

Pruning Attention or FFN

To enhance fine-tuning speed and save memory, LoRA selectively fine-tunes a subset of the large model, such as the Attention or FFN (Feed-Forward Network) layers. Additionally, in order to maintain compatibility with LoRA, we also investigate the option of fine-tuning and pruning specifically targeting the Attention or FFN layers. As shown in the Table 5, while occupying similar GPU memory, LoRAPrune demonstrates fewer parameters and competitive performance compared to LoRA. By selectively fine-tuning and pruning specific layers, we can effectively optimize the trade-off between fine-tuning speed, memory efficiency, and performance, allowing for more efficient deployment and utilization of large models.

ViT-B/16 (85.8M)	Sparsity ratio	VTAB-1k					
		Tuned / Total	GPU Mem. (M)	Natural	Specialized	Structured	Mean Acc.
Full	0%	100.00%	21280	75.9	83.4	47.6	69.0
LRP-Par (All)	50%	0.75%	15310	76.6	82.4	57.1	72.0
LPR-Par (FFN)	34%	0.63%	13119	77.3	83.3	55.7	72.1
LPR-Par (Attention)	17%	0.58%	11440	78.1	83.5	57.4	73.0
LoRA-8	0%	0.23%	9347	79.5	84.6	**60.5**	74.9
LoRA-16	0%	0.90%	9655	**79.8**	**84.9**	60.2	**75.0**

Table 5: Comparisons on VTAB-1k [ZPK+19] benchmark using ViT-B/16 pre-trained on ImageNet-21k. The sparsity ratio denotes the ratio of pruned parameters, and "Tuned/Total" denotes the fraction of trainable parameters. The best result is in **bold**, and the second-best result is underlined.

Method	Speed (s/iter)	GPU Memory (M)
Full	0.941	21280
Original grad	0.693	18933
LPR-par (Ours)	**0.598**	**15310**

Table 6: To assess the fine-tuning resources required by different methods on an RTX 3090 GPU, we conducted experiments and collected statistics on throughput and memory usage. The tests were performed with a batch size of 128, and the results were averaged over 10 runs.

Efficiency Comparison with Original Gradients

To demonstrate the efficiency of LoRAPrune in pruning and fine-tuning, we compared the fine-tuning speed and memory usage between using LoRA gradients and original parameter gradients for pruning. The results are presented in Table 6, indicating that pruning with only LoRA gradients requires less memory and achieves faster fine-tuning speed. These findings highlight the advantage of utilizing LoRA gradients in the pruning process, as it enables more efficient resource utilization and faster fine-tuning.