Light-PEFT: Lightening Parameter-Efficient Fine-Tuning via Early Pruning

Anonymous ACL submission

Abstract

Parameter-efficient fine-tuning (PEFT) has emerged as the predominant technique for fine-tuning in the era of large language models. However, existing PEFT methods still have inadequate training efficiency. Firstly, the utilization of large-scale foundation models during the training process is excessively redundant for certain fine-tuning tasks. Secondly, as the model size increases, the growth in trainable parameters of empirically added PEFT modules becomes non-negligible and redundant, leading to inefficiency. To achieve task-specific efficient fine-tuning, we propose the Light-PEFT framework, which includes two methods: Masked Early Pruning of the Foundation Model and Multi-Granularity Early Pruning of PEFT. The Light-PEFT framework allows for the simultaneous estimation of redundant parameters in both the foundation model and PEFT modules during the early stage of training. These parameters can then be pruned for more efficient training. We validate our approach on GLUE, SuperGLUE, QA tasks, and various models. With Light-PEFT, parameters of the foundation model can be pruned by over 40%, while still controlling trainable parameters to be only 25% of the original PEFT method. Compared to utilizing the PEFT method directly, Light-PEFT achieves training and inference speedup, reduces memory usage, and maintains comparable performance and the plug-and-play feature of PEFT.

1 Introduction

Large-scale pre-trained language models have demonstrated outstanding performance in various natural language processing domains (Devlin et al., 2019; Brown et al., 2020; Zhang et al., 2022; Touvron et al., 2023; OpenAI, 2023). Along with the performance improvements, the scale of model parameters continues to grow, making the cost of fine-tuning models increasingly expensive. Moreover, the practice of maintaining a separate copy of the large model for each task in conventional fine-tuning incurs substantial storage costs.

To address these challenges, parameter-efficient fine-tuning (PEFT) has been proposed: freezing most parameters of the foundation model and fine-tuning only a small number of parameters (Houlsby et al., 2019; Li and Liang, 2021; Lester et al., 2021; Hu et al., 2022), thereby reducing the computational resource requirements during training and performing nearly full-parameter fine-tuning. In addition, this technique eliminates the need to save an entire model copy for each task. During inference, task-specific models can be obtained by switching directly to the appropriate parameter-efficient module for the given task.

However, the training efficiency of existing PEFT methods still needs improvement. The first problem lies in the excessive redundancy of using a large-scale foundation model during fine-tuning for specific tasks, which results in substantial computational costs. A typical strategy is to integrate PEFT with quantization (Dettmers et al., 2023; Kim et al., 2023). Nonetheless, these methods only quantize parameters to low-bit in memory, without reducing the number of parameters and they still need to be dequantized to high-bit during training, leading to wasted training time. Another more direct approach for reducing parameters is model structured pruning (Li et al., 2022a; Hedegaard et al., 2022). However, most methods mainly focus on the inference efficiency of the model, which means they may result in higher training costs.

The second problem is that as the size of the foundation model increases, the number of parameters in added trainable modules also increases significantly. This introduces a lot of redundancy in trainable parameters, leading to inefficiency in fine-tuning. For instance, the commonly used methods LoRA (Hu et al., 2022) and QLoRA (Dettmers et al., 2023) empirically insert the low-rank modules onto fixed weight. However, there is no need

to uniformly add trainable modules of the same rank to all weights for fine-tuning each task. An improved approach is the dynamic rank method (Zhang et al., 2023; Valipour et al., 2023; Ding et al., 2023), which adaptively allocates module parameters by progressively calculating the importance of the rank during training. However, these methods require continuous estimation during training and show limited improvement in actual training efficiency.

In this paper, we introduce a novel framework named Light-PEFT, which aims to enhance the efficiency of the PEFT technique during fine-tuning. The framework consists of two methods: Masked Early Pruning of Foundation Model and Multi-Granularity Early Pruning of PEFT. In the early training stage, Light-PEFT estimates redundant parameters in both the foundation model (heads and intermediate dimensions) and the PEFT modules (module importance and rank importance) simultaneously. Structured pruning is used to eliminate this redundancy, resulting in a lighter foundation model and PEFT module for more efficient fine-tuning.

To validate the effectiveness of our Light-PEFT framework, we conduct extensive evaluations on various foundation models (RoBERTa, OPT-1.3B, OPT-6.7B), different PEFT structures (LoRA, Adapter), and on diverse benchmarks (GLUE, SuperGLUE, and question-answering tasks). The empirical results indicate that the proposed Light-PEFT framework outperforms other baseline methods in performance. It significantly improves training efficiency that reduces training memory usage by 39% and accelerates training by 1.6×. Additionally, the Light-PEFT framework improves inference efficiency that reduces inference memory by 46% and increases inference speed by 1.43×.

2 Related Works

2.1 Parameter-Efficient Fine-Tuning

Parameter-Efficient Fine-Tuning has been proposed to reduce the computational cost of fine-tuning entire model parameters (Houlsby et al., 2019; Li and Liang, 2021; Hu et al., 2022). Following works aim to further improve the efficiency of PEFT.

Improvements to the PEFT module. The motivation behind of this category of methods is that previous works often insert trainable modules empirically, resulting in uniform ranks for all inserted modules that are not parameter-efficient. AdaLoRA (Zhang et al., 2023) proposes obtaining the optimal rank for each module by iteratively pruning ranks during training. DyLoRA (Valipour et al., 2023) achieves this through dynamic training on a range of ranks. Recently, SoRA (Ding et al., 2023) introduces a mask on the ranks and gradually makes each module sparse. However, all of these methods gradually calculate the rank allocation during training, which does not improve the actual training efficiency in fine-tuning. Our method estimates the rank allocation for each module in the early stage of training and utilizes the pruned parameter-efficient modules to improve training efficiency during fine-tuning.

Improvements to the training paradigm of PEFT. To enhance training efficiency, one idea is to further reduce the memory footprint during training. QLoRA (Dettmers et al., 2023) and PEQA (Kim et al., 2023) reduce memory usage by quantizing the foundation model, while LST (Sung et al., 2022) and MEFT (Liao et al., 2023), respectively alleviate the memory footprint of intermediate activations in the foundation model through methods ladder side-tuning and reversible structures. Our approach is orthogonal to these methods from a memory perspective and can be combined with them. We explore early-stage pruning of the foundation model to reduce memory usage. Moreover, our approach can lower computational costs, speed up training, and improve inference efficiency.

Combining PEFT with pruning, most of works focus on improving inference efficiency. PST (Li et al., 2022b) and DSEE (Chen et al., 2023) propose combining unstructured pruning and PEFT, which hardly achieves acceleration on practical hardware. SPAs (Hedegaard et al., 2022) integrates structured pruning of the foundation model with PEFT, while CPET (Zhao et al., 2023) proposes distilling knowledge into PEFT modules simultaneously with pruning to reduce performance degradation. Concurrently to our works, APT (Zhao et al., 2024) reduces the training cost of the CPET method, presenting more efficient distillation and pruning. However, these methods, including APT, still require higher training time and memory costs compared to the original PEFT methods. Our approach aims to reduce the original PEFT training costs, including speed and memory, by employing early-stage structured pruning to train a non-redundant PEFT model efficiently, while improving inference efficiency simultaneously.

2.2 Structured Pruning of Models

Model pruning has been proposed to compress redundant parameters in models (LeCun et al., 1989; Kurtic et al., 2022; Liu et al., 2022; Ma et al., 2023), with structured pruning being the most straightforward method to achieve acceleration on actual hardware. For structured pruning of Transformer models, the focus lies in pruning attention heads (Michel et al., 2019), intermediate dimensions (McCarley et al., 2021), entire layers (Fan et al., 2020), and hybrid methods (Xia et al., 2022; Tao et al., 2023; Xia et al., 2024). However, most structured pruning works require additional costs during training to obtain smaller and more accurate models for inference efficiency. In terms of training efficiency, You et al. (2020) based on the lottery ticket hypothesis (Frankle and Carbin, 2019), discovered the existence of early winning tickets in DNN models, allowing early pruning to enhance subsequent training efficiency. Subsequently, Chen et al. (2021) identified early tickets in BERT models to enhance the efficiency of BERT's pre-training and fine-tuning. We follow these works and explore early pruning in parameter-efficient fine-tuning and generative foundation models.

3 Preliminaries

3.1 Parameter-Efficient Fine-Tuning

In our framework, we choose two of the most widely used methods: Adapter (Houlsby et al., 2019) and LoRA (Hu et al., 2022) to validate our approach.

Adapter. For each layer in the foundation model, including the attention sub-layer and the feed-forward sub-layer, Adapter inserts a trainable MLP module after each sub-layer. It consists of a down-projection layer $W_{down} \in R^{d \times r}$, followed by a non-linear activation function f, and finally an up-projection layer $W_{up} \in R^{r \times d}$, where d is the hidden size of the foundation model, and r is the bottleneck dimension in the trainable module, with $r \ll d$. The Adapter method can be formulated as follows:

$$h \leftarrow h + f(hW_{down})W_{up} \quad (1)$$

where h is the output of the inserted sub-layer.

LoRA. For each linear weight matrix $W \in R^{d \times d}$ in the foundation model, the LoRA method adds trainable MLP modules in parallel to W. The trainable module includes a down-projection layer W_{down} and an up-projection layer W_{up}. The LoRA method can be be formulated as follows:

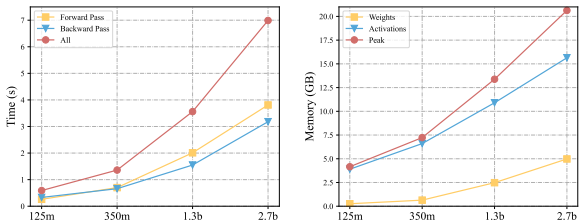
$$h \leftarrow h + s \cdot XW_{down}W_{up} \quad (2)$$

where X represents the input to the linear weight matrix W and s is a hyper-parameter scaling factor.

3.2 PEFT Training Efficiency

In this section, we present observations on the training efficiency of PEFT. We utilize LoRA to observe the results on two foundation models, RoBERTa (FP32) and OPT (FP16). For training samples, we set the length to 128 with a batch size of 32 and the time is the sum of 10 batches. All tests are conducted on a single NVIDIA RTX 3090 GPU.

The impact of foundation models size. From the perspective of training speed (Figures 1a), PEFT methods reduce the gradient computation time, so the forward pass time gradually surpasses the backward pass time. Nonetheless, the forward calculation still unchanged and needs to use all model parameters to propagate the state forward and back-propagate the loss through the entire model, becoming slower as the model size increases. From the memory perspective (Figure 1b), although PEFT techniques reduce the memory consumption of optimizer states and gradients, the model weights and intermediate activations still occupy a significant amount of memory during training. Compressing the foundation model to a smaller size can better alleviate it. This highlights the importance of reducing the parameter redundancy of the foundation model for training efficiency.

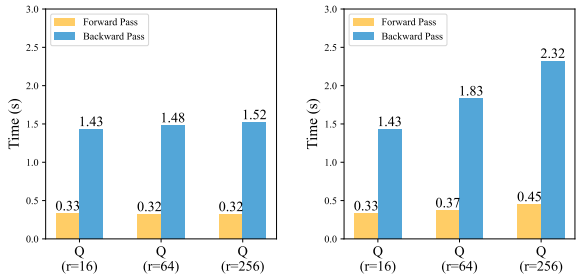


(a) Training Time (b) Memory Usage

Figure 1: Impact of foundation model size on training efficiency.

The impact of PEFT modules. We explore the impact of intra-module rank and the number of PEFT modules on training efficiency. From the perspective of training speed, Figure 2a presents experiments where we keep same modules and

only increased the rank. Figure 2b shows experiments where we keep the same trainable parameter, adding structured PEFT modules to different weights. It can be observed that when increasing the number of PEFT modules compared to varying the rank, both forward and backward times significantly increased. This indicates that, during training, the impact on speed of adding more structured PEFT modules is significantly larger than that of increasing in rank of a single structured module. From a memory perspective, the trainable parameters affect the memory consumption of optimizer states and gradients during training. As the size of the foundation model increases, the redundancy introduced by empirically adding trainable parameter modules impacts training efficiency.



(a) Module rank (b) Number of modules

Figure 2: Impact of intra-module rank and the number of PEFT modules on training speed

4 Methodology

4.1 Overview of Light-PEFT

Our goal is to eliminate parameters redundancies in the early stage, thereby reducing the computational costs of fine-tuning. Thus, we propose the Light-PEFT framework as shown in Figure 3, which consists of two methods: Masked Early Pruning of Foundation Model to reduce the redundancy of the foundation model and Multi-Granularity Early Pruning of PEFT to reduce the redundancy of the trainable parameters. First, both methods in our framework simultaneously estimate redundancies during the early-stage of training, where the total training steps are denoted as t, and the estimation for early pruning steps denoted as t', $t' \ll t$. After estimation, we prune redundancies in both, thus obtain a non-redundant foundation model and PEFT modules for more efficient training. In practice, after fine-tuning, we only need to store additional mask vectors, which are much smaller than PEFT modules, to record the pruning index of the foundation model. During the inference, our method allows the masks and PEFT modules to be easily changed, maintaining the plug-and-play feature.

4.2 Masked Early Pruning of Foundation Model

A typical Transformer model (Vaswani et al., 2017) consists of L layers, each with a multi-head attention (MHA) sub-layer and a feed-forward network (FFN) sub-layer. A MHA sub-layer contains N_H attention heads and weight matrices $W_Q^{(i)}, W_K^{(i)}, W_V^{(i)} \in \mathbb{R}^{d \times d_H}$, $W_O \in \mathbb{R}^{d \times d}$ are used for query, key, value and output, where d is the hidden size and $d_H = d/N_H$ is the hidden size of a head. In parameter-efficient fine-tuning, the weights of the foundation model are frozen, and we add the PEFT module's ΔW to these matrices. Taking LoRA module as an example, for an input X the output of the MHA is calculated as follows:

$$head^{(i)} = (W_Q^{(i)} + \Delta W_Q^{(i)}, W_K^{(i)} + \Delta W_K^{(i)}, W_V^{(i)} + \Delta W_V^{(i)}, X) \quad (3)$$

$$\text{MHA}(X) = Concat(head^{(1)}, ..., head^{(N_H)})(W_O + \Delta W_O) \quad (4)$$

To identify redundancy in attention heads, we introduce a trainable scalar mask m_A in each layer's MHA sub-layer. Now the MHA become:

$$head^{(i)} = m_A^{(i)} \cdot (W_Q^{(i)} + \Delta W_Q^{(i)}, W_K^{(i)} + \Delta W_K^{(i)}, W_V^{(i)} + \Delta W_V^{(i)}, X) \quad (5)$$

For a FFN sub-layer, which contains activation function $\text{Act}(\cdot)$ and weight matrices W_{fc1} and W_{fc2} which denote up-projection and down-projection. With PEFT modules, for an input X the output of the FFN is calculated as follows:

$$\text{FFN}(X) = \text{Act}(X(W_{fc1} + \Delta W_{fc1})) \cdot (W_{fc2} + \Delta W_{fc2}) \quad (6)$$

We also introduce a trainable scalar mask m_F in each layer's FFN sub-layer to eliminate redundancy in intermediate dimension. Now the FFN become:

$$\text{FFN}(X) = m_F \cdot \text{Act}(X(W_{fc1} + \Delta W_{fc1})) \cdot (W_{fc2} + \Delta W_{fc2}) \quad (7)$$

Inspired by Liu et al. (2017), we then use L1 regularization to learn masks m_A and m_F. During the mask learning, the PEFT module and the mask are trained jointly using gradient descent, which

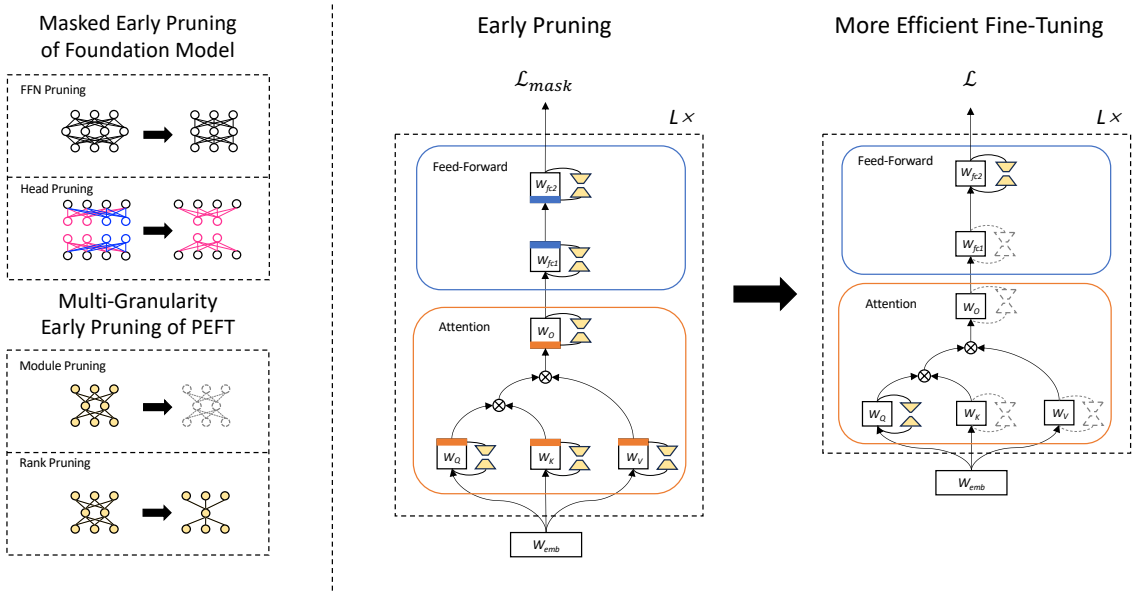


Figure 3: Illustration of Light-PEFT. The left side of the Figure shows the two methods in Light-PEFT. On the right side is an illustration of the paradigm. Firstly, both methods simultaneously estimate redundancies during the early-stage of training. After estimation, Light-PEFT prunes redundancies in both, obtaining a non-redundant foundation model and PEFT modules for more efficient training.

allowing the mask to better present the impact of PEFT to the foundation model training on the target task. The loss function is as follows:

$$\mathcal{L}_{mask} = \mathcal{L} + \lambda_A \|m_A\|_1 + \lambda_F \|m_F\|_1 \quad (8)$$

where $\mathcal{L}$ is the original loss in fine-tuning, λ_A and λ_F are hyper-parameters to control the penalty of regularization. The masks are initialized to 1 at the beginning of training.

After estimating, we perform structured pruning on attention heads with pruning ratio ρ_A layer-wise and intermediate dimensions with ρ_F globally based on the magnitudes of m_A and m_F.

4.3 Multi-Granularity Early Pruning of PEFT

In comparison to the fine-grained sparsity (i.e. rank allocation) that is the focus of most previous works (Zhang et al., 2023; Valipour et al., 2023), our preliminary observation also confirms the significance of coarse-grained module pruning for training speed. Therefore, we propose multi-granularity PEFT pruning to consider both aspects simultaneously. Furthermore, we perform pruning PEFT in the early stage to maximize efficiency during training, a technique not found in previous works.

4.3.1 Modules Pruning

To achieve coarse-grained module pruning, we begin with the original design of PEFT, where we believe that the importance of a module is primarily determined by the change it brings to the original information. Specifically, for the LoRA method, we add a trainable module $W_{down}W_{up}$ on the weight W. Thus, given an input X, the importance ratio I_M is defined as:

$$I_M = \frac{\|X \cdot W_{down}W_{up}\|_2}{\|X \cdot W\|_2} \quad (9)$$

where $\|\cdot\|_2$ represents the L2 norm, measuring the magnitude of the vector output from the PEFT module. Because one of the weight matrices in the PEFT module, such as W_{up} in the LoRA method, is typically initialized to zero. Therefore, during training, the ratio of the output magnitude of the LoRA module to the weight W's output magnitude indicates the importance of the changes required by the module added at that position.

For the Adapter method, a trainable module is added after a sub-layer. Given the output h of the previous sub-layer, the importance ratio I_M is defined as:

$$I_M = \frac{\|f(hW_{down})W_{up}\|_2}{\|h\|_2} \quad (10)$$

where I_M represents the change in information of the Adapter module on the output information h of the previous sub-layer.

In the implementation, to better estimate the importance of all added positions, for the LoRA method, we add LoRA modules on all weights of the foundation model. This may results in higher costs compared to the original LoRA in the short term, but our early estimation steps are significantly smaller than the total training steps, allowing for a substantial reduction in total costs. For the Adapter method, we follow the original approach by adding them after both the MHA and FFN sub-layers. After estimation, we use I_M to globally prune the entire PEFT modules with the pruning rate ρ_M.

4.3.2 Ranks Pruning

Not only coarse-grained pruning, we further perform fine-grained pruning on the rank of the modules. This allows us to reduce more trainable parameters and enhance training efficiency. Our motivation is based on the fact that not all modules require the same rank. To determine the importance of the rank in each module, we first calculate the importance of a single neuron, following Molchanov et al. (2017), we use first-order Taylor expansion for estimation:

$$I_{W_{ij}} = \mathbb{E}_{x\sim D}\left|\frac{\partial \mathcal{L}(x)}{\partial W_{ij}} W_{ij}\right| \tag{11}$$

where W_{ij} represents the i-th row and j-th column of neurons in W_{up} or W_{down} of the PEFT module. D represents the data used for estimation, and x denotes a batch sampled from D. The importance of the rank is determined by the sum of the importance of its connected weights, i.e., the sum of the importance of connections from neurons in W_{up} and W_{down} modules to the rank. Then, we globally prune the unimportant ranks with pruning rate ρ_R.

5 Experiments

5.1 Experimental Setup

Datasets and evaluation. We conduct experiments on eight natural language understanding (NLU) tasks from GLUE (Wang et al., 2019b) and SuperGLUE (Wang et al., 2019a) and six question-answering (QA) tasks. Because our goal is to enhance training efficiency, training on small datasets does not hold much significance. As a result, we choose four larger datasets from GLUE including MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), QQP[1], and SST-2 (Socher et al., 2013), and four larger datasets from SuperGLUE comprising ReCord (Zhang et al., 2018), WiC (Pilehvar and Camacho-Collados, 2019), BoolQ (Clark et al., 2019), and MultiRC (Khashabi et al., 2018). For MNLI, we report accuracy on the matched validation set. For QNLI, QQP, SST-2, WiC and BoolQ we report accuracy. For ReCord we report F1 and for MultiRC we report F1 over all answer-options ($F1_a$). The QA tasks including OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), ARC-Easy and ARC-Challenge (Clark et al., 2018), SciQ (Welbl et al., 2017) and WebQuestions (Berant et al., 2013). We report accuracy on all QA tasks by lm-evaluation-harness (Gao et al., 2023).

Baselines. We use Roberta-Large for NLU tasks, OPT-1.3B and OPT-6.7B for QA tasks as our foundation models. We choose several baselines to verify the effectiveness of our method. **Full-FT** is the conventional approach for fine-tuning. **Adapter** (Houlsby et al., 2019) and **LoRA** (Hu et al., 2022) are original structures we used in our framework. **LayerDrop** (Fan et al., 2020) is a strong baseline method that enhances training efficiency by dynamically dropout layers during training. We re-implement it combining with LoRA method. **LST** (Sung et al., 2022) improves model training efficiency by avoiding backpropagation in the foundation model. **Offsite-Tuning** (Xiao et al., 2023) uses a emulator derived from the foundation model for efficient training, and replaces the emulator's layers back into the foundation model for inference. **LLM-Pruner** (Ma et al., 2023) prunes model on small amount of task-agnostic corpora and restores performance using LoRA, thereby improving training efficiency. We re-implement their original task-agnostic pruning and added a task-specific pruning implementation using 1k task data.

Implementation. For the GLUE benchmark, we control the estimation steps for early pruning to be around 5% of the total training steps. For the more challenging SuperGLUE benchmark, we set the estimation steps to be within 10%. For QA tasks, we uniformly use 10% of the training steps. Please refer to the AppendixA for detailed pruning settings, as well as other training details.

5.2 Experimental Results

5.2.1 Experiments on NLU Tasks

We first evaluate our method on the GLUE benchmark. As shown in Table 1, we achieve comparable performance with the original method while using

[1]https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

Method	#Trainable Params	#Foundation Model Params	GLUE					Training Speed up
			MNLI	QNLI	QQP	SST-2	Avg.	
Full-FT	335.0M	100%	90.4	94.7	92.2	96.4	93.4	0.7×
Adapter	0.8M	100%	90.8	94.7	91.5	96.3	93.3	1×
LoRA	0.8M	100%	90.6	94.9	91.6	96.2	93.3	1×
LayerDrop	0.8M	67%	87.4	91.7	88.3	94.7	90.5	1.4×
LST	8.6M	100%	86.7	90.2	89.7	95.1	90.4	1.4×
Ours (Adapter)	0.3M	72%	88.3	93.2	**89.8**	95.6	91.7	1.4×
Ours (LoRA)	0.3M	72%	**89.4**	**93.6**	89.7	**95.9**	**92.2**	1.4×
Ours (Adapter)	0.3M	67%	87.6	93.1	89.1	95.4	91.3	**1.6**×
Ours (LoRA)	0.3M	67%	**89.0**	**93.5**	**89.2**	**95.8**	**91.9**	**1.6**×

Table 1: Results of GLUE benchmark. The training speed is measured on a single NVIDIA RTX TITAN 24GB GPU with batch size=32 and sequence length=128. Note that the speed computed here also includes the time required for estimation before pruning.

Method	#T.P.	#F.P.	SuperGLUE				
			ReCord	WiC	BoolQ	MultiRC	Avg.
Adapter	0.8M	100%	89.5	71.0	84.3	82.4	81.8
Ours	0.3M	76%	86.0	70.1	81.2	76.0	78.3
LoRA	0.8M	100%	88.3	72.7	84.1	82.7	82.0
Ours	0.3M	76%	86.6	70.2	83.3	78.0	79.5

Table 2: Results of SuperGLUE Benchmark.

72% of the foundation model parameters (pruning 5/16 of the heads and 1/3 of the FFN intermediate dimensions) and 0.3M trainable parameters by pruning PEFT modules and ranks. This results in a 1.4× training speedup and improvements in memory usage due to pruning. Furthermore, our method outperforms the baseline methods with the same speed, having fewer trainable parameters. When increasing the pruning rate and retaining 67% of the parameters in the foundation model, Light-PEFT achieves a 1.6× training speedup while still ensuring slightly better performance than the baselines. On the more challenging SuperGLUE benchmark, as shown in Table 2, we prune 4/16 of the heads and 30% of the FFN intermediate dimensions, retaining 76% of the parameters in the foundation model and 0.3M trainable parameters. This achieves performance comparable to the original PEFT method , demonstrating the effectiveness of our method Masked Early Pruning of Foundation Model.

5.2.2 Experiments on QA Tasks

For the QA tasks, we first conduct experiments on OPT-1.3B. We prune parameters (12/32 heads and 2/5 intermediate dimensions), retaining 64% and 1.5M trainable parameters, achieving comparable performance to the original method. When the trainable parameter in the original LoRA method is set to 1.5M, our method outperforms the original LoRA under fewer foundation model parameters, which demonstrates the effectiveness of our method Multi-Granularity Early Pruning of PEFT.

Compared to Offsite-Tuning, our approach achieves better performance without the high training costs of the distillation. Compared to LLM-Pruner, our method outperforms both task-agnostic and specific implementations, and our pruning process does not require the large model's gradients, leading to significantly reduced computational costs. Even with a pruning rate of 54%, we maintain better performance than the baselines.

On the larger OPT-6.7B model, pruning more foundation model parameters than OPT-1.3B and using 5.2M trainable parameters, we achieve performance comparable to the original method. When reducing trainable parameters to 2M, our method still demonstrates good performance. These experimental results demonstrate that in QA tasks, we can use the Light-PEFT framework to remove more redundant parameters from the foundation model and trainable modules, improving training efficiency with almost no loss in performance.

5.3 Analysis

5.3.1 Ablation Study

In the Section 5.2, we have demonstrated the performance of foundation model pruning (more experiments in Appendix A.2). Here, we conduct ablation study to examine two PEFT pruning strategies, module pruning and rank pruning (Table 4). Compared to not using any PEFT pruning, using module pruning or rank pruning generally improves

Method	#Trainable Params	#Foundation Model Params	QA Tasks						
			OpenBookQA	PIQA	ARC-E	ARC-C	SciQ	WebQs	Avg.
			OPT-1.3B						
Full-FT	1.3B	100%	31.4	75.2	61.3	27.7	92.5	31.2	53.2
Offsite-Tuning	-	100%	29.0	74.5	59.4	27.8	92.9	26.2	51.6
LoRA (r=64)	12.6M	100%	33.6	74.7	59.5	29.5	92.0	29.8	53.2
LoRA (r=8)	1.5M	100%	29.6	74.6	59.9	29.1	93.0	28.7	52.5
LLM-Pruner(agnostic)	10.6M	70%	29.0	72.4	54.0	24.7	89.2	20.7	48.3
LLM-Pruner(specfic)	10.6M	70%	30.4	72.9	55.9	27.6	88.7	26.5	50.3
Ours (LoRA)	1.5M	64%	33.2	74.1	59.0	28.4	92.7	28.6	52.7
Ours (LoRA)	1.9M	54%	33.2	72.6	57.6	27.5	91.8	28.2	51.8
			OPT-6.7B						
Offsite-Tuning	-	100%	33.8	77.7	66.8	33.9	91.9	23.9	54.7
LoRA(r=64)	33.6M	100%	39.2	78.5	67.5	36.7	94.0	38.5	59.1
Ours (LoRA)	5.2M	52%	39.4	74.9	63.4	32.7	92.9	35.8	56.5
Ours (LoRA)	2.0M	52%	37.2	76.0	64.4	31.7	93.3	34.7	56.2

Table 3: Results of QA Tasks.

generalization and thus enhances performance in most cases, indicating the effectiveness of the two proposed pruning strategies. Moreover, by combining the two pruning strategies, the model maintains a comparable level of performance despite having more pruned trainable parameters.

PEFT Pruning Strategy	LoRA		Adapter	
	QNLI	SST-2	QNLI	SST-2
All	93.5	95.8	93.1	95.4
w/o module p.	93.8	96.1	92.9	95.5
w/o rank p.	93.8	95.8	93.2	95.2
w/o All	93.6	95.6	93.0	95.1

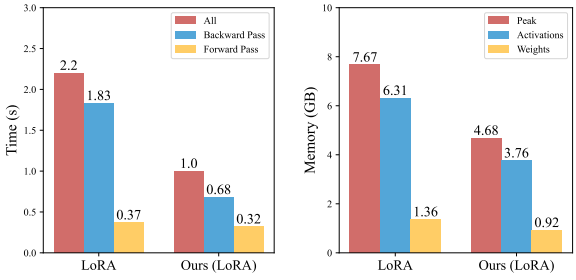
Table 4: Ablation Study of Multi-Granularity Early Pruning of PEFT.

Method	Memory	Speed
LoRA	2.51 GB	27.4 token/s
Ours (LoRA)	**1.34 GB**	**39.3 token/s**

Table 5: Inference Efficiency. We set max length=100.

5.3.2 Training and Inference Efficiency

We validate the training and inference efficiency of our method on a NVIDIA RTX 3090. In terms of training efficiency (Figure 4), we conduct experiments on RoBERTa-Large, retaining a 67% of parameters and 0.3M trainable parameters that resulted in a 32% reduction in model weight memory, a 40% reduction in activations memory, and a 39% reduction in peak memory. Calculating the total time for 10 batches, we achieve a speedup of 2.2× compared to the original LoRA. In terms of inference efficiency (Table 5), we conduct experiments on OPT-1.3B, retaining 54% of parameters and 1.9M PEFT parameters, leading to a 46% reduction in inference memory and 1.43× speed up.



(a) Training Time (b) Memory Usage

Figure 4: Training Efficiency. We set batch size=32 and sequence length=128.

6 Conclusion

This paper introduces Light-PEFT, a novel framework designed to improve the efficiency of the PEFT technique during fine-tuning. The framework comprises two methods: Masked Early Pruning of Foundation Model and Multi-Granularity Early Pruning of PEFT. The Light-PEFT framework estimates redundant parameters in both the foundation model and PEFT modules during the early stage of training and prunes them to achieve more efficient training. We validate our approach on GLUE, SuperGLUE, and QA tasks using various models. The experiments demonstrates that Light-PEFT achieves a training and inference speedup, reduced memory usage, and maintained comparable performance.

Limitations

Although Light-PEFT has achieved improved training and inference efficiency along with good performance, our work primarily focuses on the single-task fine-tuning scenario. A future direction worth exploring is the estimation and early pruning of redundant parameters on the multi-task learning scenario, enabling efficient fine-tuning across multiple tasks.

Ethics Statement

The goal of our Light-PEFT framework is to enhance training efficiency and reduce computational resource costs, which has positive impacts.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. 2021. Early-BERT: Efficient BERT training via early-bird lottery tickets. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2195–2207, Online. Association for Computational Linguistics.

Xuxi Chen, Tianlong Chen, Weizhu Chen, Ahmed Hassan Awadallah, Zhangyang Wang, and Yu Cheng. 2023. DSEE: Dually sparsity-embedded efficient tuning of pre-trained language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8208–8222, Toronto, Canada. Association for Computational Linguistics.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. 2023. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4133–4145, Singapore. Association for Computational Linguistics.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa,

Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Lukas Hedegaard, Aman Alok, Juby Jose, and Alexandros Iosifidis. 2022. Structured pruning adapters. *CoRR*, abs/2211.10155.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. 2023. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. 2022. The optimal BERT surgeon: Scalable and accurate second-order pruning for large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4163–4181, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yann LeCun, John S. Denker, and Sara A. Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 598–605. Morgan Kaufmann.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Yuchao Li, Fuli Luo, Chuanqi Tan, Mengdi Wang, Songfang Huang, Shen Li, and Junjie Bai. 2022a. Parameter-efficient sparsity for large language models fine-tuning. In *31th International Joint Conference on Artificial Intelligence*.

Yuchao Li, Fuli Luo, Chuanqi Tan, Mengdi Wang, Songfang Huang, Shen Li, and Junjie Bai. 2022b. Parameter-Efficient Sparsity for Large Language Models Fine-Tuning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 4223–4229. International Joint Conferences on Artificial Intelligence Organization.

Baohao Liao, Shaomu Tan, and Christof Monz. 2023. Make pre-trained model reversible: From parameter to memory efficient fine-tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yuanxin Liu, Fandong Meng, Zheng Lin, Peng Fu, Yanan Cao, Weiping Wang, and Jie Zhou. 2022. Learning to win lottery tickets in BERT transfer via task-agnostic mask training. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5840–5857, Seattle, United States. Association for Computational Linguistics.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2755–2763. IEEE Computer Society.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

J. S. McCarley, Rishav Chakravarti, and Avirup Sil. 2021. Structured pruning of a bert-based question answering model.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14014–14024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. LST: Ladder side-tuning for parameter and memory efficient transfer learning. In *Advances in Neural Information Processing Systems*.

Chaofan Tao, Lu Hou, Haoli Bai, Jiansheng Wei, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2023. Structured pruning for efficient generative pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10880–10895, Toronto, Canada. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287, Dubrovnik, Croatia. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations*.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1513–1528, Dublin, Ireland. Association for Computational Linguistics.

Guangxuan Xiao, Ji Lin, and Song Han. 2023. Offsite-tuning: Transfer learning without full model. *CoRR*, abs/2302.04870.

Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Yingyan Lin, Zhangyang Wang, and Richard G. Baraniuk. 2020. Drawing early-bird tickets: Toward more efficient training of deep networks. In *International Conference on Learning Representations*.

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

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *CoRR*, abs/1810.12885.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068.

Bowen Zhao, Hannaneh Hajishirzi, and Qingqing Cao. 2024. APT: adaptive pruning and tuning pretrained language models for efficient training and inference. *CoRR*, abs/2401.12200.

Weilin Zhao, Yuxiang Huang, Xu Han, Zhiyuan Liu, Zhengyan Zhang, and Maosong Sun. 2023. CPET: effective parameter-efficient tuning for compressed large language models. *CoRR*, abs/2307.07705.

A Appendix

A.1 Details of Experimental Setup

Hardware. We use NVIDIA RTX TITAN for GLUE for experiments, NVIDIA RTX 3090 for SuperGLUE experiments and experiments using the OPT-1.3B Model in QA Tasks, NVIDIA A800 for experiments using the OPT-6.7B Model in QA Tasks.

Hyper-parameters. We use AdamW as the optimizer for training. Other detailed settings for NLU tasks are provided in Table 6, while the settings for QA tasks can be found in Table 7 and Table 8.

A.2 The impact of the pruning rate on the foundation model.

We analyzed the impact of different foundation model pruning rates on performance on the WiC dataset (Figure 5). It was observed that within a certain range (above 62.5%), pruning resulted in a relatively minor decrease in performance. However, once this threshold was exceeded, a significant performance decline occurred, demonstrating that pruning within this range removes redundant parameters.

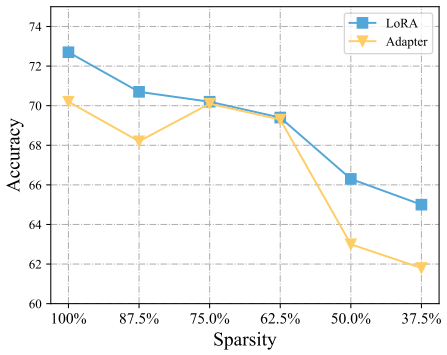


Figure 5: The impact of the pruning rate on the foundation model.

A.3 The impact of the estimation steps of early pruning

We analyzed the impact of the early pruning estimation steps on performance using the BoolQ dataset (Figure 6). It was observed that once the estimation steps exceeded 6.8% of the total training steps, further estimation did not lead to performance improvement. This demonstrates that our method can effectively identify redundant parameters in both the foundation model and PEFT modules during the early stage of training.

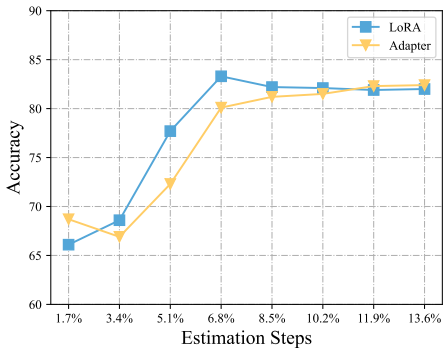


Figure 6: The impact of the estimation steps of early pruning.

Method	Dataset	MNLI	QNLI	QQP	SST-2	ReCord	WiC	BoolQ	MultiRC
LoRA	Estimation Steps	1000	1000	1000	800	2000	340	400	600
	Rank	8							
	ρ_M	75%							
	ρ_R	50%							
	Estimation lr	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4
	Fine-Tuning lr	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4
	Batch Size	32	32	32	32	32	16	32	16
	Sequence Length	128	128	128	128	256	128	128	384
	# Epochs	5	5	5	10	5	50	20	20
Adapter	Estimation Steps	1000	1000	1000	800	2000	340	400	1000
	Rank	8							
	ρ_M	25%							
	ρ_R	50%							
	Estimation lr	6e-4	8e-4	3e-4	6e-4	6e-4	3e-4	6e-4	7e-4
	Fine-Tuning lr	4e-4	3e-4	3e-4	3e-4	3e-4	1e-4	6e-4	5e-4
	Batch Size	32	32	32	32	32	16	32	16
	Sequence Length	128	128	128	128	256	128	128	384
	# Epochs	5	5	5	10	5	50	20	20

Table 6: Hyperparameters for NLU Tasks.

Method	Dataset	OpenBookQA	PIQA	ARC-E	ARC-C	SciQ	WebQs
LoRA	Estimation Steps	1 Epoch	1 Epoch	1 Epoch	1 Epoch	1 Epoch	1 Epoch
	Rank	8					
	ρ_M	50%/50%					
	ρ_R	50%/25%					
	Estimation lr	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4
	Fine-Tuning lr	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4
	Batch Size	64	64	64	64	64	64
	Sequence Length	128	128	128	128	128	128
	# Epochs	10	10	10	10	10	10

Table 7: Hyperparameters for QA Tasks on OPT-1.3B.

Method	Dataset	OpenBookQA	PIQA	ARC-E	ARC-C	SciQ	WebQs
LoRA	Estimation Steps	1 Epoch	1 Epoch	1 Epoch	1 Epoch	1 Epoch	1 Epoch
	Rank	8					
	ρ_M	50%/75%					
	ρ_R	25%/50%					
	Estimation lr	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4
	Fine-Tuning lr	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4
	Batch Size	64	64	64	64	64	64
	Sequence Length	128	128	128	128	128	128
	# Epochs	10	10	10	10	10	10

Table 8: Hyperparameters for QA Tasks on OPT-6.7B.