

Efficient Mask Learning for Language Model Fine-Tuning

Minping Chen

The Hong Kong University of Science
and Technology (Guangzhou)
Guangzhou, China
mchen779@connect.hkust-gz.edu.cn

Ruijia Yang

The Hong Kong University of Science
and Technology (Guangzhou)
Guangzhou, China
ryang379@connect.hkust-gz.edu.cn

Zeyi Wen*

The Hong Kong University of Science
and Technology (Guangzhou)
Guangzhou, China
wenzeyi@hkust-gz.edu.cn

ABSTRACT

Parameter-efficient fine-tuning (PEFT) of pre-trained language models (PLMs) has shown promising results by updating significantly fewer parameters than full fine-tuning. Masking-based fine-tuning is one type of PEFT methods, by freezing the majority of the model parameters during fine-tuning. Existing masking-based fine-tuning methods either need to manually select the trainable parameters (heuristic-based), or perform mask learning to adaptively select the trainable parameters with high memory and computation cost. To address these problems, this paper proposes Low-Rank based Efficient Mask Learning (LoReML). LoReML performs mask learning based on low-rank decomposition and matrix reconstruction with a small ratio of new parameters. After mask learning, LoReML uses the scaled intermediate results in mask learning as warm start initialization to boost the model quality, then freezes the masked parameters accordingly, and fine-tunes the PLM. Moreover, LoReML exploits sparse training techniques to enhance the memory efficiency in masking-based fine-tuning. Experimental results across various tasks and pre-trained backbones demonstrate that LoReML can notably outperform existing heuristic-based methods. Moreover, LoReML achieves competitive or better performance compared with the adaptive mask learning methods, while improving memory and computation efficiency by over 50% in mask learning.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing; Neural networks.**

KEYWORDS

Parameter-efficient Fine-tuning; Language Model Fine-tuning

ACM Reference Format:

Minping Chen, Ruijia Yang, and Zeyi Wen. 2025. Efficient Mask Learning for Language Model Fine-Tuning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25)*, November 10–14, 2025, Seoul, Republic of Korea. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3746252.3761178>

*Corresponding Author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM '25, November 10–14, 2025, Seoul, Republic of Korea

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-2040-6/2025/11

<https://doi.org/10.1145/3746252.3761178>

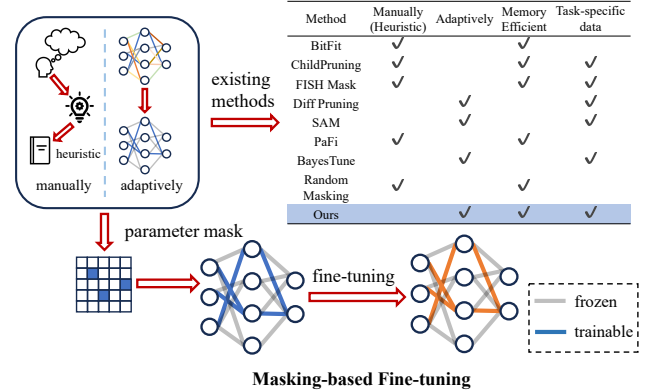


Figure 1: Illustration of masking-based fine-tuning and comparison of existing methods.

1 INTRODUCTION

Pre-trained language models (PLMs), especially Large-scale Language Models (LLMs), have achieved remarkable performance on various natural language processing tasks through full fine-tuning on task-specific data [4, 27, 33]. However, with the continuous growth of the sizes of language models, full fine-tuning is resource-consuming, restricting their wide applications to multi-task learning and resource-constrained scenarios.

To address this issue, parameter-efficient fine-tuning (PEFT) has emerged as an effective solution for adapting PLMs to specific tasks by only updating a small number of parameters. In addition to improving the fine-tuning efficiency, models fine-tuned with PEFT can share the same base model and only need to store a small number of task-specific parameters for each task, and thus reducing the memory cost in model deployment/serving. Most existing PEFT methods can be categorized into two groups: adapter-based fine-tuning and masking-based fine-tuning. Adapter-based fine-tuning introduces new modules or parameters to the PLMs and only the newly added parameters/modules are trainable [14, 15, 22]. Different from adapter-based methods, masking-based fine-tuning selects a small subset of the existing model parameters to be trainable [10, 23, 35], as illustrated in Figure 1.

Existing masking-based fine-tuning methods either need to select the subset of trainable parameters manually [23, 40, 42], or perform mask learning with high memory and computation cost [10, 18]. As an example of manual selection of trainable parameters, BitFit [42] chooses the bias terms of the PLM to be the trainable parameters. For the example of high memory and computation cost, Diff Pruning [10] requires introducing the same number of new parameters as

the PLM for mask learning, which is even more memory-consuming and computationally expensive than full fine-tuning. To avoid manually defining the mask and the high cost in mask learning, this paper proposes Low-Rank based Efficient Mask Learning (LoReML). Different from previous methods, LoReML only needs to add a small fraction of new parameters by low-rank decomposition, instead of introducing the same number of new parameters as the PLM and fully fine-tuning the model. During the forward pass of mask learning, LoReML reconstructs the mask matrix based on the low-rank matrices, and subsequently combines the pre-trained weight matrix and the reconstructed one with scaling for proper integration. When finished learning, LoReML transforms the reconstructed matrix into a binary mask matrix according to the top k absolute values which tend to contribute most greatly to the gradients.

The proposed LoReML offers the following benefits for fine-tuning: (i) It avoids the hand-crafted efforts of selecting the components of the model to insert new modules or parameters compared to the adapter-based methods, and avoids manually defining the masks compared to the heuristic-based methods. (ii) It is more promising to select a better subset of parameters for fine-tuning according to the learning process of the specific task, thus enhancing the model quality. (iii) Its model quality is further boosted with a warm start initialization by leveraging the reconstructed low-rank matrices, i.e., adding them to the pre-trained weights with careful scaling. Compared to existing adaptive mask learning methods which perform the second-stage fine-tuning with the pre-trained model as initialization, LoReML has already learned relevant features and patterns for the specific task from the first stage. (iv) It saves substantial memory consumption compared to full fine-tuning with an efficient implementation. This efficient implementation exploits data sparsity for fine-tuning, i.e., the trainable parameters and their gradients are stored in sparse format.

The main contributions of this paper are as follows.

- (1) We propose an efficient mask learning method called LoReML based on low-rank decomposition and matrix reconstruction, which improves memory efficiency and reduces computational costs compared with existing methods.
- (2) We exploit a warm start initialization for the second-stage fine-tuning using the intermediate results in mask learning. This warm start can boost both the model quality and convergence rate.
- (3) We demonstrate the effectiveness of LoReML with extensive experiments across various tasks and pre-trained backbones. The results indicate that LoReML significantly surpasses current heuristic-based methods. Furthermore, LoReML delivers performance that is on par with or better than existing adaptive mask learning methods, while halving the memory and computation cost in mask learning.

2 METHODOLOGY

In this section, we present the technical details of our Low-Rank based Efficient Mask Learning (LoReML) method. For completeness, we first introduce some background of masking-based fine-tuning. Then, we provide an overview of LoReML, and elaborate on its details of mask learning in the first stage, fine-tuning with warm start and its efficient implementation in the second stage.

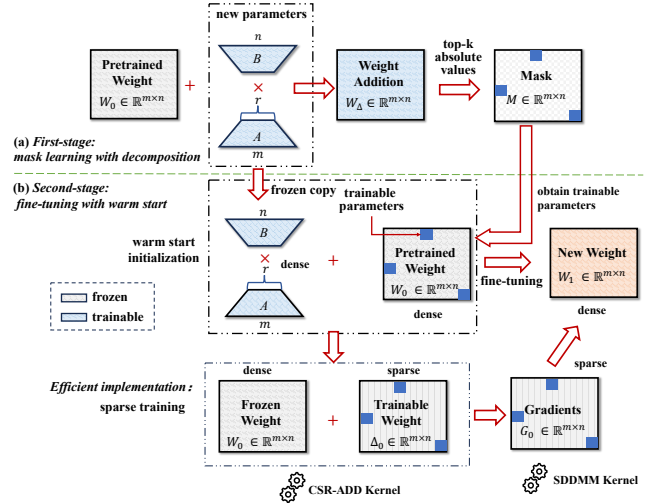


Figure 2: The overview of our LoReML method.

2.1 Masking-based Fine-tuning

Masking-based fine-tuning first determines which parameters of the pre-trained language model (PLM) should be trainable, and a binary mask is constructed heuristically or automatically by learning. Existing methods either (i) use heuristically-motivated rules to generate the parameter mask [23, 40, 42], or (ii) adaptively learn the mask through fine-tuning (i.e., learnable mask) [8, 10, 18]. Adaptive mask learning is more promising for selecting a better subset of parameters for fine-tuning compared with heuristic selection. However, previous adaptive mask learning often requires introducing 100% new parameters [10, 18] or performing full fine-tuning for the PLM [2, 8], which is inefficient in terms of memory and computation. In this paper, we focus on developing a more memory and computation efficient method for adaptive mask learning. Before providing the technical details of our method, we define the common notations in the following.

Let $W_0 \in \mathbb{R}^{m \times n}$ be a pre-trained weight matrix of the PLM, $M \in \{0, 1\}^{m \times n}$ be the binary mask, where 1s in M indicate that the corresponding parameters are trainable. M is often a sparse matrix with only a small number of 1s. After the mask M is constructed, the PLM can be fine-tuned, with only a selection of the parameters being trainable according to the mask. For mask learning, generally, the PLM is fully fine-tuned and the masks can be obtained according to some criterion based on the weight addition, i.e., the difference between the pre-trained weight and the fine-tuned weight:

$$W_1 = W_0 + W_\Delta, \quad (1)$$

where W_1 denotes a weight matrix of the fine-tuned model, and W_Δ is its weight addition after fine-tuning. Then the model is fine-tuned again, by only updating a small ratio of the parameters based on the sparse mask.

2.2 Overview of LoReML

Here, we present an overview of our LoReML method, as shown in Figure 2. It aims to relieve the handcrafted work of defining the

parameter masks, while addressing the memory and computation efficiency problems of previous methods for adaptive mask learning. LoReML has two stages including mask learning with efficient decomposition and fine-tuning with a warm start. In the first stage, the low-rank decomposition for the mask matrix is introduced for adaptive mask learning. Due to the low-rank decomposition, only a small number of new parameters are needed in LoReML, resulting in higher memory efficiency when storing and updating the weight addition matrix during mask learning compared to the existing methods. Along with the improvement in memory consumption, our method enjoys a lower computation cost in the backward pass as well, because LoReML only needs to update the low-rank matrices instead of the whole mask matrix. In the second stage, LoReML performs fine-tuning with a warm start by exploiting the weight addition matrix, i.e., the reconstructed matrix, learned in the first stage to initialize the backbone model. This warm start can boost both the convergence rate of fine-tuning and the final model quality on the specific task, since the weight addition matrix consists of knowledge learned in the first stage. Besides, we implement the second stage of fine-tuning efficiently with sparse training techniques by leveraging two sparse operation kernels. Next, we describe the technical details of mask learning and fine-tuning in LoReML.

2.3 Mask Learning with Decomposition

To address the inefficiency problem in memory and computation of existing methods, our proposed LoReML method learns the masks by low-rank decomposition, as shown at the top of Figure 2. Hence, LoReML introduces only a small number of new parameters to represent the weight addition \mathbf{W}_Δ . For a pre-trained weight matrix $\mathbf{W}_0 \in \mathbb{R}^{m \times n}$, the weight addition can be reconstructed based on the low-rank decomposition matrices according to Equation (2) below.

$$\mathbf{W}_0 + \frac{\alpha}{r} \mathbf{W}_\Delta = \mathbf{W}_0 + \frac{\alpha}{r} \mathbf{AB}, \quad (2)$$

where $\mathbf{A} \in \mathbb{R}^{m \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times n}$ are two low-rank decomposition matrices; α is a constant that is regarded as a hyper-parameter and the rank $r \ll \min(m, n)$. \mathbf{W}_Δ needs to be 0 at the beginning of mask learning, so we use a random Gaussian initialization for \mathbf{B} and zero for \mathbf{A} following [15]. Thus, $\mathbf{W}_\Delta = \mathbf{AB}$ is zero before mask learning. Besides, \mathbf{W}_Δ is scaled by $\frac{\alpha}{r}$, since this scaling assists in minimizing the necessity of returning hyper-parameters when changing the value of r [41]. During mask learning, \mathbf{W}_0 is frozen as shown in the grey part in Figure 2(a) while \mathbf{A} and \mathbf{B} are trainable as shown in the blue part in Figure 2(a). The forward pass is rewritten from $\mathbf{H} = \mathbf{W}_0 \mathbf{X} + b$ to $\mathbf{H} = (\mathbf{W}_0 + \frac{\alpha}{r} \mathbf{AB}) \mathbf{X} + b$. When the model finishes mask learning, the reconstructed weight addition \mathbf{W}_Δ can be transformed into the mask \mathbf{M} which is a binary matrix. As our goal is mask learning for the model parameters, we add the low-rank decomposition matrices to all the weight matrices of the model. Note that vectors of bias terms and normalization layers are not included, since they cannot be decomposed into smaller matrices so we keep them frozen during mask learning.

When the masking learning is completed, we can obtain a binary mask for each parameter of the model (except the bias and the normalization layer) according to the weight addition \mathbf{W}_Δ . Although other binarization approaches [23, 35] can be used in our proposed LoReML method, the top k absolute values are transformed into

1s as they tend to contribute most greatly to the gradients during mask learning, while the rest are transformed into zeros. There are different ways to calculate the top k absolute values, i.e., the group top k and the global top k . For the group top k , each weight matrix is regarded as a group, while for the global top k , all parameters of the model are treated as a group. LoReML can easily incorporate these different types of top k choices of masks. As our main goal is to address the high memory and computation cost issues of existing mask learning methods, we uniformly set the ratio of trainable parameters for each layer similar to existing methods.

2.4 Fine-tuning with Warm Start

2.4.1 Fine-tuning with Warm Start. Once we obtain the masks, LoReML can move on to the second stage which is fine-tuning with a warm start. Different from previous methods [8, 10, 18] which use the pre-trained weights as the initialization and discard the weight addition learned from the first stage, we initialize the PLM utilizing the weight addition \mathbf{W}_Δ learned from the first stage. In particular, for each pre-trained weight matrix \mathbf{W}_0 , we combine it and its weight addition \mathbf{W}_Δ to get a warm start for fine-tuning. With such a warm start, the model can leverage task-specific knowledge learned from the first stage, and thus the performance of the model can be enhanced. Besides, this warm start can lead to faster convergence because the model has already learned useful representations for the specific task. Note that in the first stage, the \mathbf{W}_Δ is scaled by $\frac{\alpha}{r}$, so here we also apply this scaling factor, as shown below:

$$\mathbf{W}_{\text{init}} = \mathbf{W}_0 + \frac{\alpha}{r} \mathbf{W}_\Delta, \quad (3)$$

where \mathbf{W}_{init} is the warm start initialization of the second stage of fine-tuning. Different from the first stage for mask learning where \mathbf{W}_Δ can be updated based on the updating of the low-rank matrices, in this stage, \mathbf{W}_Δ is used for warm start initialization and most of the parameters in it are frozen (only the selected parameters are trainable). Finally, we can perform masking-based fine-tuning. Compared with full fine-tuning which updates all the parameters of the model, we only update the selected parameters based on the learned masks.

2.4.2 Efficient Implementation. Most previous masking-based fine-tuning methods compute all gradients of the model parameters like full fine-tuning, and multiply the gradients with the masks to set the gradients of the frozen parameters to zeros [2, 23, 35]. This implementation leads to even more memory consumption than full fine-tuning due to the need to store the gradient masks. To address this issue, we provide an efficient implementation of our method inspired by a sparse matrix operation method [30]. With this sparse implementation, we can achieve the goal of selectively computing the gradients of the trainable parameters (i.e., sparse gradients) rather than computing the gradients of all parameters (i.e., dense gradients). This is achieved by leveraging a CSR-ADD kernel and a Sampled Dense-Dense Matrix Multiplication (SDDMM) kernel [30]. The CSR-ADD kernel calculates the sum of a dense matrix and a sparse matrix and the SDDMM kernel multiplies two dense matrices where only specific elements of the output are required (i.e., the output is sparse).

To improve memory efficiency, the forward step and the backward step are implemented as follows. As shown in Figure 2, in the

Table 1: Trainable parameters and memory comparison during mask learning (i.e., the first stage).

Method	%Tuned	RoBERTa-base		RoBERTa-large	
		#Param.	Memory	#Param.	Memory
Diff Pruning	200%	250.0M	12.7GB	710.0M	32.3GB
SAM/BayesTune	200%	250.0M	13.3GB	710.0M	37.1GB
LoReML	~ 0.5%	0.65M	5.5GB	1.99M	13.2GB

forward step, for a dense pre-trained weight \mathbf{W}_0 which is frozen, we add it with a trainable sparse matrix Δ_0 where the non-zero values indicate the trainable parameters of \mathbf{W}_0 . Then the output \mathbf{O} is obtained by $\mathbf{O} = \mathbf{X}(\mathbf{W}_0 + \Delta_0)$, where \mathbf{X} is the input. For the corresponding backward step, according to the chain rule, the gradients of \mathbf{X} and Δ_0 are $\frac{\partial \mathcal{L}}{\partial \mathbf{X}} = \frac{\partial \mathcal{L}}{\partial \mathbf{O}} (\mathbf{W}_0 + \Delta_0)^\top$ and $\frac{\partial \mathcal{L}}{\partial \Delta_0} = \mathbf{X}^\top \frac{\partial \mathcal{L}}{\partial \mathbf{O}}$ respectively. Note that $\frac{\partial \mathcal{L}}{\partial \Delta_0}$ is still sparse by using the SDDMM kernel. Through this sparse implementation, we can save significant memory compared with full fine-tuning and the dense implementation in most previous masking-based methods, as they need to compute and store all gradients as well as all optimizer states.

3 ANALYSIS OF MASK LEARNING

In this section, we compare the number of trainable parameters, memory consumption, and computation costs between previous methods and LoReML for mask learning.

3.1 Memory Cost for Mask Learning

We present the ratio of trainable parameters comparison and memory comparison during mask learning of different methods in Table 1. The baselines introduce the same number of new parameters as the pre-trained weights, and update both the pre-trained weights and the newly introduced parameters. Therefore, their tuned 200% model parameters. In contrast, LoReML only needs to update about 0.5% parameters for mask learning, which intuitively shows the parameter efficiency of our proposed LoReML method. Additionally, we test the memory consumption using a batch size of 32 and a sequence length of 128 for all methods. LoReML can reduce more than 50% memory consumption compared with the baselines on both RoBERTa-base and RoBERTa-large, and the higher memory efficiency of LoReML still holds for larger models.

3.2 Computation Analysis of Mask Learning

We also present a computation cost comparison of mask learning between the baselines shown in Table 1 and LoReML. As the baselines all perform full fine-tuning for mask learning, their computation cost is equally likely. Specifically, we compute the number of floating-point operations (FLOPs) of a learning step during mask learning of different methods. A learning step consists of a forward pass and a backward pass, and most PLMs contain an embedding layer and multiple transformer layers. We summarize the FLOPs of different operations in a single layer as well as the total FLOPs of a single training step of the baselines and LoReML in Table 2, where r is the rank, e is the embedding dimension, v is the vocabulary size, s is the sequence length, h is the hidden size, N_{head} is the number of attention heads, and i is the dimension of the intermediate

Table 2: FLOPs comparison of different operations in a learning step of a single layer (using a batch size of 32 and a sequence length of 128).

Operation	Baselines	LoReML
Token Embed.	$6evs$	$4evs + 4rs(e + v)$
Other Embed.	$6es(s + 3)$	$4es^2 + 10es + 4rs(e + s + 2)$
QKV Proj.	$9s(2h^2)$	$6s(2h^2) + 24hrs$
Att. Output	$6h^2s$	$4h^2s + 8hrs$
Intermediate	$6his$	$4his + 4s(hr + ri)$
Output	$6his$	$4his + 4s(hr + ri)$
Others	$12hs^2 + 30s^2N_{head} + 75hs + 27s(e + i)$	
One step FLOPs	3.74×10^{13}	2.60×10^{13}

Table 3: Runtime for mask learning on RoBERTa-large.

Diff Pruning	BayesTune	SAM	LoReML
~ 3h30min	~ 3h40min	~ 1h40min	~ 40min

layer. For simplicity, we only list the operations with different computational costs for the baseline methods and our method, while merging those with identical computational costs into the *Others* category, including the operations of layer normalization, dropout, bias, residual addition, and activation function, etc.

According to the work of Kaplan et al. [17], the FLOPs of the backward pass is approximately two times that of the forward pass for training a Transformer model. Thus, we formulate the FLOPs of a forward pass to approximately obtain the FLOPs of a backward pass as two times that of its forward pass for the baselines. Compared with the baselines, LoReML requires additional forward computation due to the addition weight matrix reconstruction which involves the multiplication of the two low-rank matrices \mathbf{A} and \mathbf{B} . However, in the backward pass, LoReML can significantly reduce FLOPs as it only updates the parameters of the low-rank matrices while the baselines need to update the pre-trained weights and the newly introduced parameters. The overall FLOPs of a forward and a backward pass during mask learning is the sum of the FLOPs of the embedding layer and multiple Transformer layers, shown as:

$$\text{FLOPs} = (\text{FLOPs}^{(E)} + L \times \text{FLOPs}^{(T)}) \times S, \quad (4)$$

where $\text{FLOPs}^{(E)}$ is the FLOPs of the embedding layer, $\text{FLOPs}^{(T)}$ is the FLOPs of a single Transformer layer, L is the number of Transformer layers and S is the total training steps. Note that the FLOPs of the classifier is omitted in Equation (4) as it remains unchanged for different methods.

Based on Equation (4), taking RoBERTa-large as the backbone, and setting the batch size to 32 and the sequence length to 128, the FLOPs of a single training step of the baselines and our LoReML method are about 3.14×10^{13} and 2.60×10^{13} respectively. Our method is theoretically more efficient than existing methods for mask learning. Furthermore, we present the actual runtime of different methods for mask learning, considering that the convergence rate of each method is different, as shown in Table 3. LoReML demonstrates a runtime for mask learning that is over 50% shorter than that of other baselines.

Table 4: Performance of different masking-based methods on the GLUE benchmark using RoBERTa-large as the backbone. We denote the best scores in bold and underline the second best results. [†] indicates scores sourced from the work of Chen *et al.* [2023] and Liao *et al.* [2023].

Method	%Tuned	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
Zero-shot	0%	43.0	39.1	49.6	62.5	0.0	-	68.1	52.7	45.0
Full FT [†]	100%	90.2	92.2	<u>94.7</u>	<u>96.4</u>	68.0	<u>92.4</u>	90.9	86.6	88.9
BitFit [†]	0.1%	88.0	90.2	93.4	96.1	64.2	90.9	92.7	86.2	87.7
Diff Pruning [†]	0.5%	90.3	90.3	94.6	96.4	65.1	92.0	90.2	84.5	87.9
FISH Mask [†]	0.5%	90.2	89.8	94.1	96.1	66.3	92.5	88.7	86.3	88.0
SAM	0.5%	89.7	89.8	94.1	96.1	64.3	92.3	89.5	87.4	87.9
BayesTune	0.5%	89.7	89.5	93.1	95.4	65.8	92.4	89.7	86.3	87.7
RM	0.5%	90.2	91.3	94.3	96.0	63.1	91.2	89.5	87.0	87.8
PaFi [†]	0.5%	90.2 _{0.05}	90.3 _{0.05}	94.6 _{0.08}	96.7 _{0.19}	70.2 _{0.46}	91.9 _{0.24}	91.4 _{0.14}	88.8 _{0.63}	89.3
LoReML	0.5%	90.4 _{0.05}	91.7 _{0.05}	94.6 _{0.13}	<u>96.6</u> _{0.12}	<u>69.1</u> _{1.19}	92.6 _{0.08}	<u>92.1</u> _{1.03}	89.7 _{0.75}	89.6

4 EXPERIMENTS AND DISCUSSIONS

4.1 Datasets and Models.

Datasets. We evaluate our method on: (i) the GLUE benchmark [39], which is extensively used in previous works; (ii) generation tasks including XSum [29], WMT 2016 en-ro [3], OpenBookQA [28], and the MMLU benchmark [13]. Task details are listed below.

- Natural language inference: *MNLI*, *QNLI*, *RTE*
- Sentiment analysis: *SST-2*; Question answering: *QQP*
- Textual similarity: *STS-B*; Paraphrase detection: *MRPC*
- Linguistic acceptability: *CoLA*
- Abstractive summarization: *XSum*
- English to Romanian translation: *WMT 2016 en-ro*
- Open book question answering: *OpenBookQA*
- Instruction tuning covering 57 subjects: *MMLU*, including 5 categories: STEM, humanities, social sciences, and others.

Following previous works [5, 15], we use Matthews correlation to evaluate *CoLA*, Spearman correlation to evaluate *STS-B*, and accuracy to evaluate other GLUE datasets. As the test sets of the GLUE benchmark are not publicly available, we report the results on the development sets, following recent works [15, 23]. For XSum, we report ROUGE-1/2/L scores [24] on the test set. For WMT 2016 en-ro and OpenBookQA, we report the BLEU scores [31] and accuracy on their test sets, respectively. For the MMLU benchmark, we fine-tune the model on the Alpaca instruction dataset [36], and evaluate the five-shot accuracy on this benchmark.

Evaluation models. We evaluate LoReML on the GLUE benchmark using RoBERTa-large [27] as the backbone. For the generation tasks, BART-large [21] and its multilingual version mBART-large [26] are selected as the backbone models to evaluate the summarization task and translation task, respectively. In addition, we evaluate the summarization task on OPT-6.7B [43], LLaMA-7B and LLaMA-13B [38]. However, as the OPT model and the LLaMA model are pre-trained mainly using English corpora, they are not suitable for learning English to Romanian translation tasks. Therefore, we evaluate our method with another generation task (i.e., question task) using the OPT model and the LLaMA model as the backbone, respectively. Additionally, we evaluate the MMLU benchmark on LLaMA-7B, LLaMA-13B and Qwen-2.5 7B [37].

4.2 Experimental Setup

Baselines. LoReML compares with these masking-based baselines:

- (1) **Zero-shot**: it evaluates the PLM without fine-tuning using manual prompts adapted from Gao et al. [9].
- (2) **Full FT**: it updates all parameters during fine-tuning.
- (3) **BitFit** [42]: it only updates the bias terms in the PLM.
- (4) **Diff Pruning** [10]: it first learns masks with L_0 -norm penalty approximation to encourage sparsity, then it is fine-tuned with a small set of trainable parameters.
- (5) **FISH Mask** [35]: it uses the Fisher information to estimate the importance of each parameter and selects the top ones as trainable parameters.
- (6) **SAM** [8]: it proposes a second-order approximation method to determine the tunable parameters by optimizing the approximation function.
- (7) **BayesTune** [18]: it uses a sparse Laplace prior for each parameter to select the important parameters adaptively.
- (8) **PaFi** [23]: it selects the bottom k parameters to be trainable according to their absolute magnitudes.
- (9) **Random Masking (RM)** [40]: it randomly selects a ratio of trainable parameters in each layer’s query and value matrix.

Most of the existing masking-based methods have not conducted experiments on large language models, e.g., LLaMA-7B and 13B. Due to the limited computation resources, we only select the two most recent methods, i.e., PaFi and RM, as baselines for experiments on larger models and run the experiments for them.

Implementations. For the GLUE benchmark, following previous studies [5, 23], we set the ratio of trainable parameters of the baselines and our method to 0.5%, except for BitFit which is set to 0.1% since it only updates the bias terms. We report the average results of five runs with different random seeds. For the low-resource tasks (i.e., *RTE*, *MRPC* and *STS-B*), we use the fine-tuned model of the *MNLI* task as the initial model rather than the original PLM following previous methods [15, 23, 27].

Compared with the GLUE tasks, the generation tasks require more trainable parameters to achieve competitive results as full fine-tuning. Therefore, we set the ratio of trainable parameters of the baselines and our method to 5% when using BART-large as the

Table 5: Performance comparison on the generation tasks. We report the average results of three random seeds, denote the best scores in bold and underline the second best results.

BART-large		XSum			WMT en-ro
Method	%Tuned	ROUGE-1	ROUGE-2	ROUGE-L	BLEU
Full FT	100%	44.8	21.9	36.8	37.3
BitFit	0.1%	40.6	17.3	32.2	26.4
PaFi	5%	44.1	20.7	35.7	36.8
LoReML	5%	44.8	21.4	36.4	36.9
		XSum			OpenBookQA
Method	%Tuned	ROUGE-1	ROUGE-2	ROUGE-L	Accuracy
OPT-6.7B					
Zero-shot	0%	17.2	2.1	12.7	24.6
Full FT	100%	42.5	18.0	35.0	76.1
RM	0.5%	35.6	13.8	28.2	74.9
PaFi	0.5%	42.7	18.1	35.1	76.3
LoReML	0.5%	42.9	18.3	35.3	76.7
LLaMA-7B					
Zero-shot	0%	18.4	2.3	13.5	12.4
Full FT	100%	<u>43.9</u>	<u>19.3</u>	36.2	84.5
RM	0.5%	30.8	11.7	24.2	83.0
PaFi	0.5%	43.8	19.2	36.1	82.4
LoReML	0.5%	44.1	19.6	36.4	85.9
LLaMA-13B					
Zero-shot	0%	18.5	2.3	13.5	0.4
Full FT	100%	<u>45.4</u>	<u>21.0</u>	<u>37.8</u>	90.2
PaFi	0.5%	45.3	20.9	37.7	86.2
LoReML	0.5%	45.7	21.3	38.0	87.2

backbone. However, due to the limited resources, we still set the ratio of trainable parameters to 0.5% on larger models, i.e., OPT-6.7B, LLaMA-7B, LLaMA-13B and Qwen-2.5 7B. Besides, we only use 10k randomly selected instances of the XSum training data for fast evaluation, while on BART-large, we use the whole training set for fine-tuning. Due to the limited space, for the hyper-parameters used in our experiments, please refer to our code¹.

4.3 Main Results

Results on the GLUE benchmark. The predictive accuracy of different masking-based methods on the GLUE benchmark using RoBERTa-large as the backbone model is summarized in Table 4. LoReML achieves the best performance on average evaluation score among different masking-based methods. Besides, LoReML achieves the best performance on five out of eight datasets, with up to 1.4% improvement in predicted accuracy. For the remaining three datasets, LoReML achieves the second-best performance. Another observation is that LoReML achieves a higher average score than Full FT. This is because the GLUE benchmark contains classification or regression tasks that are relatively simple, and our method with 0.5% trainable parameters can already fit these tasks.

Results on the generation tasks. We present the performance comparison on the generation tasks in Table 5 and Table 6. LoReML outperforms PaFi and RM in all metrics across various generation tasks and pretrained backbones, consistently showing the effectiveness of LoReML. There is a significant performance gap between RM and LoReML on the XSum dataset, showing that random masking

Table 6: Performance comparison on the MMLU benchmark.

Method	Humanities	STEM	Social Sciences	Other	Avg.
LLaMA-7B					
Full FT	38.6	34.0	45.3	47.6	41.0
PaFi	34.6	32.0	39.5	41.7	36.7
LoReML	37.6	33.8	43.2	45.7	39.8
LLaMA-13B					
Full FT	47.3	39.2	56.7	56.0	49.5
PaFi	45.6	36.9	54.5	54.7	47.6
LoReML	47.0	37.6	55.2	55.1	48.5
Qwen-2.5 7B					
Full FT	66.9	70.6	83.6	76.7	73.6
PaFi	67.6	70.1	83.0	75.9	73.4
LoReML	67.9	71.0	83.8	77.0	74.1

Table 7: Performance comparison with other PEFT methods on the GLUE benchmark and the MMLU benchmark.

Model	Method	%Tuned	GLUE Avg. Accuracy
RoBERTa-base /large	Prefix	0.5%	84.6/85.7
	Adapter	0.5%	84.4/88.1
	LoRA	0.5%	86.4/88.8
	LoReML	0.5%	87.6/89.6
Model	Method	%Tuned	MMLU Avg. Accuracy
LLaMA-7B	LoRA	0.5%	37.8
	LoReML	0.5%	39.8
Qwen-2.5 7B	LoRA	0.5%	69.5
	LoReML	0.5%	74.1

cannot work very well on complex tasks. Besides, LoReML achieves more significant improvements on the MMLU benchmark (e.g., 3.1% in average accuracy on LLaMA-7B), which is an instruction-following benchmark.

4.4 More Detailed Inspection of LoReML

Performance comparison beyond masking-based methods. In addition to comparing LoReML with existing masking-based methods, we also compare it with other types of PEFT methods, e.g., some popular adapter-based methods including Prefix [22], Adapter [14] and LoRA [15]. We present the average accuracy comparison on the GLUE benchmark and the MMLU benchmark of different methods in Table 7. LoReML achieves significant performance improvements compared with these strong adapter-based methods across various pretrained backbones. Among these adapter-based baselines, LoRA is related to LoReML since it uses low-rank decomposition for mask learning. LoReML identifies the parameters that most benefit from fine-tuning for the specific tasks, enhancing the model’s ability to generalize and adapt to them. In comparison, LoRA does not employ a mechanism to discriminate which parameters are important to update. It is more like an approximation of full fine-tuning, increasing the risk of overfitting. Besides, the warm start initialization in LoReML also helps to improve the model performance.

¹Code and other implementation details are available at: <https://github.com/Chan1996/LoReML>.

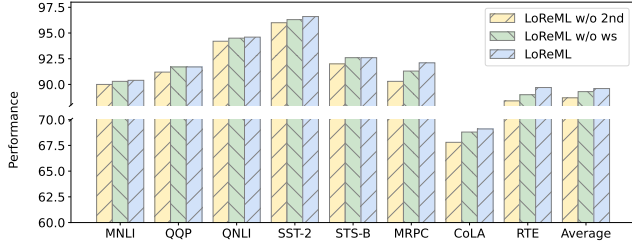


Figure 3: Ablation studies on the GLUE benchmark.

Table 8: Memory usage comparison in the second stage.

Model	Method	#GPU	BS/GPU	Total Memory
LLaMA-7B	Full FT	4 × A800	8	315.7GB
	Adapter	2 × A800	16	128.0GB
	LoRA	2 × A800	16	128.6GB
	LoReML	2 × A800	16	130.9GB
LLaMA-13B	Full FT	4 × A800	2	251.3GB
	Adapter	2 × A800	8	133.6GB
	LoRA	2 × A800	8	140.8GB
	LoReML	2 × A800	8	135.4GB

Effect of different stages of LoReML. Here, we investigate the effect of different fine-tuning stages of LoReML on the GLUE benchmark using RoBERTa-large as the backbone, and the results are shown in Figure 3. Specifically, LoReML-w/o 2nd denotes the model learned in the first stage aiming for adaptive mask learning, without further fine-tuning (i.e., second stage); LoReML-w/o ws denotes a variant of LoReML which uses the original PLM as the initial model without using our proposed warm start initialization.

LoReML-w/o 2nd, LoReML-w/o ws and LoReML achieve 88.7, 89.3 and 89.6 average scores on the GLUE benchmark, respectively. Besides, LoReML outperforms LoReML-w/o 2nd on all tasks, indicating the effectiveness of the second-stage fine-tuning by leveraging the learned masks and the warm start initialization.

Second stage memory usage of LoReML. Here, we present the memory usage comparison of the second-stage fine-tuning between our method and other methods in Table 8. With the efficient implementation introduced in section 2.4, our method can save substantial memory usage compared with full fine-tuning. For example, full fine-tuning requires four A800 GPUs with a batch size of two, while our method requires only two GPUs with a batch size of eight for LLaMA-13B. Additionally, our method achieves comparable memory usage with Adapter and LoRA, while significantly outperforming them on multiple tasks (cf. Table 7). In this memory testing, we do not use other memory-saving techniques, such as gradient checkpointing and CPU offloading. However, our method can easily combine with these memory-saving techniques to further improve memory efficiency.

Convergence of LoReML. The experimental results in Figure 4 show that the proposed warm start initialization for fine-tuning yields performance enhancements on most tasks compared with the model that uses the pre-trained weights as initialization. Here,

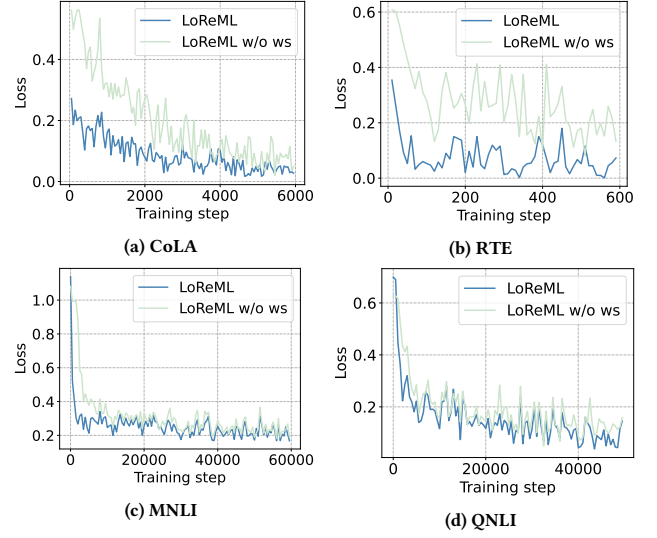


Figure 4: Convergence comparison on four tasks.

we provide a convergence comparison during fine-tuning between the two models LoReML and LoReML without warm start (“-w/o ws” for short) on four tasks of the GLUE benchmark, i.e., CoLA, RTE, MNLI and QNLI in Figure 4. From the figure, we can see that the loss of LoReML is lower than that of LoReML-w/o ws at the early stage of fine-tuning on these tasks, especially on the CoLA task and the RTE task. Furthermore, using the warm start initialization, the fine-tuning process of LoReML is more stable than that of LoReML-w/o ws. Therefore, leveraging a warm start for fine-tuning can boost the model performance on these tasks.

Analysis of the learned masks. Here, we perform some analysis on the learned masks. First, we compute the row and column sparsity across different layers of a subset of learned masks, i.e., the ratio of rows or columns whose values are all zeros, and we call it the empty ratio. In more detail, we compute this empty ratio for different tasks and different pre-trained backbones, as shown in Figure 5. We observe that models with different pre-trained backbones present different row and column sparsity of the learned masks. For RoBERTa-large which is mainly used for classification tasks, high row and column sparsity is exhibited across various layers. The possible reason is that classification tasks are relatively simple, and thus a large proportion of neurons (a column of a pre-trained weight can be seen as a neuron) can remain unchanged. For generation tasks which are more complex, the sparsity of rows and columns is much lower, especially the row sparsity, as shown in Figure 5 (c) and (d).

To assess the quality of the learned masks, we use the masks learned through full fine-tuning as a benchmark and calculate the overlap between these masks and those learned by other methods with RoBERTa-large as the backbone. As presented in Figure 6, we can see that our method achieves the highest overlap among different methods, suggesting the enhanced quality of our learned masks, and thus achieving better performance on various tasks.

Impact of hyper-parameters. Here, we investigate the impact of two important hyper-parameters on the final performance, i.e.,

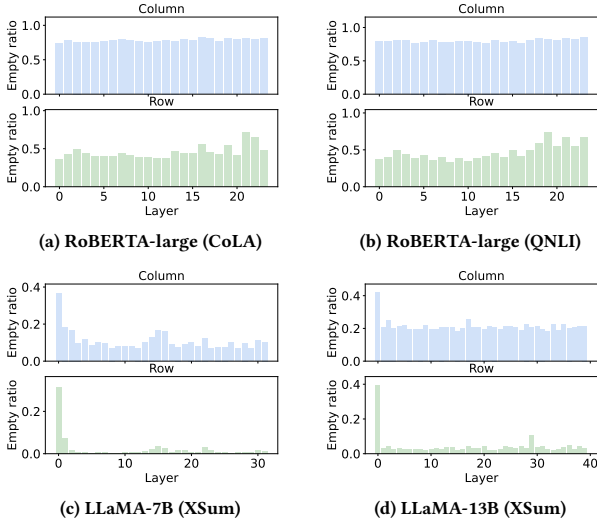


Figure 5: Row and column sparsity across different layers of a subset of learned masks.

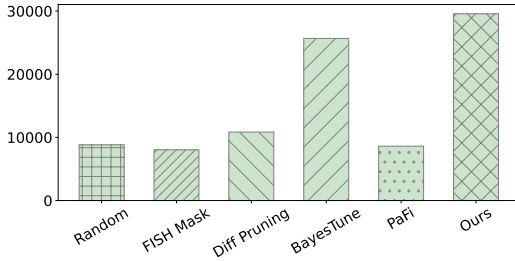


Figure 6: Mask Overlap Analysis.

the rank in mask learning and the scaling factor in the warm start initialization. Specifically, we conduct this experiment on two tasks using RoBERTa-large as the backbone, as shown in Figure 7. In our main experiments, we set the rank according to the ratio of trainable parameters of our baselines for fair comparison, which is four in this case. It is observed that using smaller ranks for mask learning may lead to performance drops, e.g., on the RTE task. This is reasonable since using smaller ranks results in more information loss in mask learning.

We also run the second-stage fine-tuning with different scaling factors $\{0.5r, 1.0r, 1.5r, 2.0r\}$ on these two tasks. The results are shown in Figure 7 (b). We can see that there is no significant performance difference using different scaling factors, indicating the performance of our method is not sensitive to the choice of scaling factors. By default, we can set the scaling factor equal to the rank r .

Impact of inconsistent tasks in the first and second stages. Our method performs mask learning adaptively for the specific tasks, i.e., the mask learning stage (first stage) and the second stage are for the same task. Here, we explore the performance when we use inconsistent tasks in the first and second stages, and the

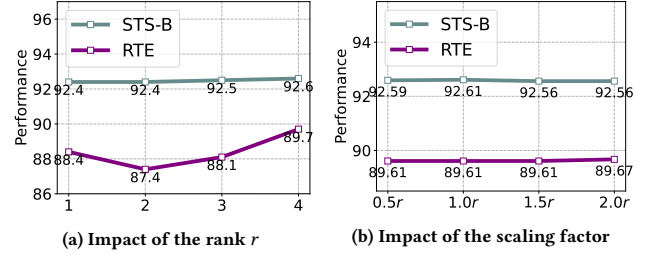


Figure 7: Sensitivity to the rank in the mask learning and scaling factor in the warm start initialization.

Table 9: Performance comparison of inconsistent and consistent tasks in the first and second stages using LLaMA-7B. For the inconsistent scenario, the first stage uses the XSum task, and the second stage uses the MMLU benchmark.

	Humanities	STEM	Social Sciences	Other
Inconsistent	34.6	31.6	38.6	41.7
Consistent	37.6	33.8	43.2	45.7

results are shown in Table 9. Compared to the model with consistent tasks in the first and second stages, the performance degradation of the model with inconsistent tasks is 2.2%~4.6%, demonstrating the acceptable domain adaptability of our method. Simultaneously, this study also suggests that adaptive mask learning for the specific task is more effective and is essential to achieve better final performance.

T-test analysis. To demonstrate how significantly our method outperforms the baselines, we conduct a t-test on the experimental results. Note that as the experimental results on the GLUE benchmark of many baselines are obtained from previous work, we lack their results of different random seeds to compute the confidence interval for each dataset. Therefore, we conduct the t-test on the generation experiments as we run the experiments for the baselines. The results are shown in Table 10. We can see that our method significantly outperforms the baselines in most cases with p-values < 0.05 . The T-test results demonstrate that our method is effective for various tasks and pre-trained backbones.

5 RELATED WORK

In this section, we discuss different groups of PEFT methods: adapter-based methods, masking-based methods, and hybrid methods that combine ideas from multiple PEFT methods.

Adapter-based fine-tuning. The first type of adapter-based method introduces an extra component called “prefix”. A representative is Prefix Tuning [22] which proposes to prepend a prefix to the input sequence and only the prefix parameters are trainable. Similar methods include Prompt Tuning [20] and P-Tuning [25]. The second type of adapter-based method adds adapters within the model. For example, Houlsby et al. [14] propose to add an adapter module twice to each Transformer layer. AdapterFusion [32] takes as input the learned representation of multiple adapters trained on different tasks and learns a parameterized mixed encoded information. LoRA [15] decomposes an added trainable weight matrix

Table 10: T-test of LoReML. ✓ denotes the p-value < 0.05, which means LoReML significantly outperforms the corresponding method. ✗ denotes the p-value > 0.05, which means the improvement of LoReML is not significant.

Model	Baseline	XSum	WMT en-ro	
BART-large	BitFit	✓	✓	
	PaFi	✓	✓	
Model	Baseline	XSum	OpenBookQA	
OPT-6.7B	RM	✓	✓	
	PaFi	✓	✓	
Model	Baseline	XSum	OpenBookQA	MMLU
LLaMA-7B	RM	✓	✓	-
	PaFi	✓	✓	✓
LLaMA-13B	PaFi	✓	✓	✓
Qwen-2.5 7B	LoRA	-	-	✓
	PaFi	-	-	✓

into a product of two low-rank matrices. Chen et al. [6] propose Hadamard Adapter, which only acts on self-attention outputs in PLMs through adopting element-wise linear transformation based on the Hadamard product. Lei et al. [19] propose CoDA which improves the inference efficiency of existing adapter tuning methods with no accuracy loss using conditional computation. Song et al. [34] introduces MultiLoRA, a paradigm that pre-trains a universal model and fine-tunes it across multiple domain divisions, integrating domain-specific patterns with a LoRA fusion module. Due to the limited space, we only discuss some representative methods.

Despite the notable success, existing adapter-based methods fail to automatically determine where to insert the newly added modules, making it require much manual effort to investigate the performance of different insert ways. Our method can automatically learn the trainable parameter subset, avoiding such manual effort.

Masking-based fine-tuning. Some masking-based methods use heuristically-motivated rules to select the trainable parameters set. One simple method is BitFit [42], which only updates the bias terms of the model. FISH Mask [35] uses the Fisher information [1, 7] to estimate the importance of each parameter and the most important ones (i.e., those with the largest Fisher information values) are selected to be trainable. Different from FISH Mask, the work of Liao et al. [23] hypothesizes that the important parameters of a PLM learn more generic knowledge and should be fixed. Thus, they only optimize the unimportant ones selected according to their absolute magnitudes. Xu and Zhang [40] propose Random Masking which applies a randomly produced binary mask on the model parameters and only updates the unmasked parameters. Other masking-based methods first perform mask learning, and then fine-tune the model with a small number of trainable parameters. LT-SFT [2] fully fine-tunes the model and takes parameters with the top k greatest absolute difference between the original PLM and the fine-tuned model as the trainable parameters. In comparison, Diff Pruning [10] introduces learnable binary masks on the weights by fine-tuning with a regularization objective which is a differentiable approximation to the L_0 norm. SAM [8] employs

a second-order approximation function to obtain an analytical solution for mask learning. BayesTune [18] proposes to use a sparse Laplace prior for each parameter to select the important parameters adaptively. These methods involve high memory and computation costs for mask learning as they fully fine-tune the model.

In summary, existing masking-based methods construct the masks manually or perform mask learning at a high cost. Our method can both avoid manual work in mask construction and perform mask learning efficiently.

Hybrid methods for fine-tuning. The hybrid methods manually design or search proper combination of PEFT approaches for a PLM. For instance, SparseAdapter [12] introduces a *Large-Sparse* strategy for tuning the adapter layers. In this strategy, a large hidden dimension is assigned for the inserted adapter, and a pruning mask is computed to prune around 40% of the values at initialization. S^3 Delta-M [16] develops a differentiable delta tuning structure search method which explicitly controls the number of trainable parameters and searches for an optimal combination of multiple existing PEFT techniques (e.g., Adapter-LR, Bitfit and LoRA, etc). The work of He et al. [11] presents a unified framework that establishes connections between existing PEFT methods, but their method is manually designed and applied uniformly to various layers of the PLM. Chen et al. [5] introduce design spaces for PEFT which allows automatic customization of both tuning structures and strategies, but their method involves sampling hundreds of models to search the optimal combination of multiple PEFT methods.

Similar to existing masking-based methods, these hybrid methods require manual design work for the proper combination of multiple PEFT methods or the high cost of automatic searching of optimal combinations. Our method is automatic and efficient in selecting the trainable parameters for fine-tuning.

6 CONCLUSION

In this paper, we have proposed a Low-Rank based Efficient Mask Learning (LoReML) method, which has two stages. In the first stage, we have proposed to learn the parameter masks by only involving a small number of trainable parameters based on low-rank decomposition and matrix reconstruction, thus improving memory and computation efficiency compared with existing methods. In the second stage, we utilized the reconstructed matrix of the low-rank matrices learned in the first stage as a warm start for fine-tuning, thereby enhancing the model quality. Additionally, we employ sparse training techniques to achieve an efficient implementation for masking-based fine-tuning. The extensive experimental results on diverse tasks and pre-trained backbone models have shown the effectiveness of LoReML and its significantly higher memory and computation efficiency in mask learning.

ACKNOWLEDGMENTS

This work is supported by the Guangzhou Industrial Information and Intelligent Key Laboratory Project (No. 2024A03J0628). It is also funded by the NSFC Project (No. 62306256) and the Natural Science Foundation of Guangdong Province (No. 2025A1515010261). Additionally, we appreciate the assistance of Mr. Yuebin Xu in running some experiments.

GENAI USAGE DISCLOSURE

In the research stage, as this paper proposes an efficient mask learning method for language model fine-tuning, our experiments are conducted on some generation language models, including LLaMA-7B/13B and Qwen-2.5 7B, et al, as introduced in Section 4.2. In the writing stage, we only use GenAI for light editing of the text, e.g., automating grammar checks and word autocorrect.

REFERENCES

- [1] Shun-ichi Amari. 1996. Neural learning in structured parameter spaces-natural Riemannian gradient. *Advances in neural information processing systems* 9 (1996).
- [2] Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulic. 2022. Composable Sparse Fine-Tuning for Cross-Lingual Transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2022. 1778–1796.
- [3] Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation (wmt16). In *First conference on machine translation*. Association for Computational Linguistics, 131–198.
- [4] 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*.
- [5] Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. 2023. Parameter-Efficient Fine-Tuning Design Spaces. In *The Eleventh International Conference on Learning Representations, ICLR 2023*.
- [6] Yuyan Chen, Qiang Fu, Ge Fan, Lun Du, Jian-Guang Lou, Shi Han, Dongmei Zhang, Zhixu Li, and Yanguhua Xiao. 2023. Hadamard adapter: An extreme parameter-efficient adapter tuning method for pre-trained language models. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM 2023*. 276–285.
- [7] Ronald A Fisher. 1922. On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character* 222, 594-604 (1922), 309–368.
- [8] Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. 2023. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 12799–12807.
- [9] Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making Pre-trained Language Models Better Few-shot Learners. 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)*, ACL/IJCNLP 2021. 3816–3830.
- [10] Demi Guo, Alexander M. Rush, and Yoon Kim. 2021. Parameter-Efficient Transfer Learning with Diff Pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers)*. 4884–4896.
- [11] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a Unified View of Parameter-Efficient Transfer Learning. In *The Tenth International Conference on Learning Representations, ICLR 2022*.
- [12] Shwai He, Liang Ding, Daizhe Dong, Jeremy Zhang, and Dacheng Tao. 2022. SparseAdapter: An Easy Approach for Improving the Parameter-Efficiency of Adapters. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 2184–2190.
- [13] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. [n.d.]. Measuring Massive Multitask Language Understanding. In *The Ninth International Conference on Learning Representations, ICLR 2021*.
- [14] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morroni, 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*, Vol. 97. 2790–2799.
- [15] 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*.
- [16] Shengding Hu, Zhen Zhang, Ning Ding, Yadao Wang, Yasheng Wang, Zhiyuan Liu, and Maosong Sun. 2022. Sparse structure search for delta tuning. *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems, NeurIPS 2022*. 35 (2022), 9853–9865.
- [17] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).
- [18] Minyoung Kim and Timothy Hospedales. 2023. BayesTune: Bayesian sparse deep model fine-tuning. *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems, NeurIPS 2023*, 36 (2023).
- [19] Tao Lei, Junwen Bai, Siddhartha Brahma, Joshua Ainslie, Kenton Lee, Yanqi Zhou, Nan Du, Vincent Zhao, Yuexin Wu, Bo Li, et al. 2023. Conditional adapters: Parameter-efficient transfer learning with fast inference. *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems, NeurIPS 2023*, 36 (2023).
- [20] 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, EMNLP 2021*. 3045–3059.
- [21] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*. 7871–7880.
- [22] 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, ACL/IJCNLP 2021, (Volume 1: Long Papers)*. 4582–4597.
- [23] Baohao Liao, Yan Meng, and Christof Monz. 2023. Parameter-Efficient Fine-Tuning without Introducing New Latency. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2023. 4242–4260.
- [24] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.
- [25] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT Understands, Too. *CoRR abs/2103.10385* (2021). <https://arxiv.org/abs/2103.10385>
- [26] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics* 8 (2020), 726–742.
- [27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR abs/1907.11692* (2019). <http://arxiv.org/abs/1907.11692>
- [28] 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, EMNLP 2018*. 2381–2391.
- [29] Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don’t Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*. 1797–1807.
- [30] Mahdi Nikdan, Soroush Tabesh, and Dan Alistarh. 2024. Rosa: Accurate parameter-efficient fine-tuning via robust adaptation. *arXiv preprint arXiv:2401.04679* (2024).
- [31] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics, ACL 2002*. 311–318.
- [32] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021*. 487–503.
- [33] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.
- [34] Zijian Song, Wenhan Zhang, Lifang Deng, Jiandong Zhang, Kaigui Bian, and Bin Cui. 2024. MultiLoRA: Multi-Directional Low Rank Adaptation for Multi-Domain Recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024*. 2148–2157.
- [35] Yi-Lin Sung, Varun Nair, and Colin Raffel. 2021. Training Neural Networks with Fixed Sparse Masks. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*. 24193–24205.
- [36] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

- [37] Qwen Team. 2024. Qwen2.5: A Party of Foundation Models. <https://qwenlm.github.io/blog/qwen2.5/>
- [38] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* abs/2302.13971 (2023). <https://doi.org/10.48550/arXiv.2302.13971>
- [39] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *7th International Conference on Learning Representations, ICLR 2019*.
- [40] Jing Xu and Jingzhao Zhang. 2024. Random Masking Finds Winning Tickets for Parameter Efficient Fine-tuning. *arXiv preprint arXiv:2405.02596* (2024).
- [41] Greg Yang and Edward J Hu. 2020. Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522* (2020).
- [42] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022*. 1–9.
- [43] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2023. Opt: Open pre-trained transformer language models, 2022. URL <https://arxiv.org/abs/2205.01068> 3 (2023), 19–0.