ShareLoRA: Less Tuning, More Performance for LoRA Fine-tuning of LLMs

Anonymous ACL submission

Abstract

Due to the prohibitively expensive full fine-tuning costs of large language models (LLMs), various popular parameter-efficient fine-tuning (PEFT) methods have been developed. These methods majorly rely on fine-tuning few add-on modules, popularly referred to as *adapters*, that corresponds to only *small fraction of LLM parameters*. In specific, low rank adaptation (LoRA), has demonstrated impressive parameter efficiency while yielding performance close to the full fine-tuning. However, classical LoRA may still fine-tune more parameters as compared to the inherent rank of the pre-trained weights (Aghajanyan et al., 2020), leaving room for further parameter reduction. To mitigate this, only recently, few researches had proposed various freezing strategy of LoRA projection matrices, however, at the cost of additional FLOPs. To improve fine-tuning efficiency, in this work, we present SHARELORA, that leverages a novel approach to use the redundancy in pre-trained model weights and share LoRA modules to significantly reduce the trainable parameter counts. In specific, SHARELORA automatically finds the redundancy of the pre-trained weights and determines which LoRA adapters can share parameters. For this, SHARELORA uses the similarity between representations to assess the information redundancy and a greedy algorithm to maximize the sharing of LoRA modules. We performed extensive evaluations with LLaMA family LLMs across various tasks. In specific, at reduced PEFT parameter count of up to **23**%, SHARELORA performs similar or better that of the existing PEFT alternatives.

1 Introduction

Large language models (LLMs), *e.g.,* GPT-4 and LLaMA2, are at the forefront of advances in the field of machine learning (ML). These large models are pre-trained on vast datasets (*e.g.,* images or text corpora) and are subsequently fine-tuned for specialized tasks, demonstrating proficiency in domains such as natural language, image processing, and fundamental scientific discoveries (Bommasani et al., 2021; Touvron et al., 2023a; Singhal et al., 2022, 2023). Foundation models (FMs), such as LLaMA2, Falcon, and MPT, are typically referred to as the *base model*. They are pre-trained solely to predict the next token to generate from their entire vocabulary space (Touvron et al., 2023a; Penedo et al., 2023; Team, 2023). To employ the base model for real applications, *e.g.,* building chatbots, these models must be further fine-tuned (*e.g.,* on multi-turn human-human or human-chatbot conversations) to either follow specific human instructions or align with human preferences (Leike et al., 2018; Ziegler et al., 2019; Chung et al., 2022).

The massive parameter scales of FMs make fine-tuning a computationally daunting task. For instance, the GPU memory capacity must be sufficiently large to host the fine-tuning of the entire set of FM parameters using a reasonably large batch size. PEFT methods have been proposed that allow for the fine-tuning of only a small fraction of FM parameters or merely a few small add-on adapters atop the FM pre-trained model backbone, all the while keeping the majority of the base FM parameters frozen (Houlsby et al., 2019; Hu et al., 2021; Zaken et al., 2021; Zhang et al., 2023). These PEFT methods democratize FM fine-tuning across a wide range of computing platforms, such as commodity hardware. LoRA, which is one of the most popular PEFT methods, has been shown to effectively reduce the GPU memory requirement during FM fine-tuning (Hu et al., 2021). LoRA achieves parameter efficiency by adding low-rank adapters in parallel with specific FM parameters, such as MLP layers or the query, key, and value parameter weights in multi-head attention. During fine-tuning, LoRA focuses solely on these low-rank adapters, leaving the large FM parameters untouched.

However, the approach of LoRA is relatively

FT Method	GSM8K ↑	ARC-Challenge ↑	WinoGrande ↑	Hellaswag ↑
LoRA($r = 12$)	**37.98**	**48.21**	**64.25**	**51.96**
Naive-shared LoRA	37.23	47.95	62.83	48.38

Table 1: Accuracy comparison of LoRA and a naive-share LoRA (that shares weights naively) strategy.

straightforward and does not take into account the importance and similarity of each layer, lacking finer control over the LoRA modules. Currently, there are some improvements to LoRA, such as AdaLoRA (Zhang et al., 2023), which dynamically adjusts the rank based on importance scores during finetuning, DoRA (Liu et al., 2024a), which separates weights into direction and magnitude for more personalized finetuning, and LoRA+ (Hayou et al., 2024), which adjusts the learning rates of A and B separately to achieve better finetuning results. However, they all overlook the redundancy in pretrained foundation models, where some layers exhibit similar behaviors. These redundancy layers can share parameters during finetuning to further reduce the memory requirements for fine-tuning.

We find that sharing the weights of the LoRA module does not significantly impact performance while reducing the number of trainable parameters. Specifically, we have experimented with sharing the LoRA weights of odd-numbered layers with those of even-numbered layers, effectively halving the number of trainable parameters. As shown in Table 1, even using this simple method of weight sharing results in only a minimal performance degradation. However, there are likely more sophisticated methods for weight sharing that could further optimize this approach. However, identifying which layers to share the weights of the LoRA module with others is challenging. This difficulty arises because the explainability of foundation models (FMs) (Zhao et al., 2024) is still an active area of research, and there is no definitive conclusion about which layers exhibit similar behaviors.

We have observed that in some current foundation models, there are cases where layer representations are similar. Due to the lack of sufficient data during training, not every parameter can learn unique information, leading to the extraction of similar features and redundant representations. Consequently, some works have used this observation to perform structural compression of the model (Gromov et al., 2024). In our approach, we share LoRA weights among redundancy layers during finetuning. By significantly reducing the number of distinct LoRA weights, we can slightly increase the LoRA rank, thereby greatly improving the finetuning performance without increasing the number of training parameters.

Our SHARELORA method consists of two main components: (i) computing similarity matrices between representations of layers and (ii) sharing the LoRA module parameters among redundancy layers. This method identifies layers with similar representations and shares their weights, reducing the number of trainable parameters while maintaining model performance. By leveraging layer similarities, SHARELORA significantly improves finetuning efficiency. We conduct extensive experiments on a wide range of tasks and models to demonstrate the effectiveness of SHARELORA. Specifically, we evaluate the performance using LLaMA-7B, LLaMA2-7B, and LLaMA3-8B for natural language commonsense reasoning and LLava-1.5-7B for image-text understanding. The results show that SHARELORA performs similar or better than LoRA, while only using 80% trainable parameter.

The summary of our contributions is as follows:

- We propose SHARELORA, a novel parameter-efficient fine-tuning (PEFT) method that leverages the similarity of layer representations to enable weight sharing, achieving this without introducing any additional train or inference latency compared to LoRA.
- We develop a greedy algorithm to determine which layers should share the weights of the LoRA module based on the similarity of their representations.
- We conduct extensive experiments demonstrating that SHARELORA consistently performs similar or better LoRA across various tasks, while using less trainable parameters.

2 Background and Motivation

Parameter Efficient of Fine Tuning. A primary research trajectory aimed at reducing the fine-tune parameters of pretrained FMs is to model the incremental updates of pretrained weights in a parameter-efficient manner. For example, given a pretrained weight matrix W, diff pruning (Guo et al., 2021) initializes Δ as the same dimensions as W and performs magnitude-based pruning on Δ. Diff pruning characterizes Δ as the incremental updates of W, and it can improve the parameter efficiency due to the sparsity of Δ. However, it requires specific hardware support to accelerate the computation of unstructured sparse matrices. This hardware-specific dependency underscores a crucial consideration for the practical deployment of such an approach in real-world applications. In addition, it does not significantly reduce computational cost compared to full fine-tuning (Hu et al., 2021), as every entry of Δ needs to be updated and then be pruned.

To tackle those limitations, Hu et al. propose LoRA (Hu et al., 2021), which parameterize Δ as the product of two low-rank matrices:

$$W' = W + \Delta = W + BA, \tag{1}$$

where $\Delta \in \mathbb{R}^{d_1 \times d_2}$, $B \in \mathbb{R}^{d_1 \times r}$, and $A \in \mathbb{R}^{r \times d_2}$ with $r \ll \{d_1, d_2\}$. As the rank r is much smaller than the dimension of W (e.g., $r = 8$ and $d_1 = d_2 = 4096$), the number of trainable parameters and training overhead dramatically decreases.

However, LoRA has its limitations, as it typically applies separate parameters for each B and A by default. This approach overlooks the significant variation in the redundancy of weight matrices across different layers and modules during the fine-tuning of pretrained foundation models. We will demonstrate this issue later. Consequently, LoRA cannot adaptively share the LoRA modules among the redundancy layers, which could otherwise achieve comparable performance with fewer trainable parameters.

Weight Sharing. Although PEFT methods can reduce the number of trainable parameters, thereby decreasing GPU memory footprint during fine-tuning, the number of parameters in LLMs also scales rapidly, making it increasingly challenging to conduct even PEFT on commodity GPUs. In this paper, we explore the possibility of combining LoRA with *weight sharing*, a method that allows multiple neural network layers to share the same model weights. Weight sharing has been a widely adopted technique to reduce the number of trainable parameters while maintaining performance and sometimes helping mitigate overfitting (Press and Wolf, 2017; de Lhoneux et al., 2018; Lan et al., 2020; Dai et al., 2020; Takase and Kiyono, 2021).

Similarity Across Layers. Our motivation for weight sharing among LoRAs across layers comes from the observation that the representations among model layers/blocks attain high levels of similarity, especially for bottom layer blocks (as

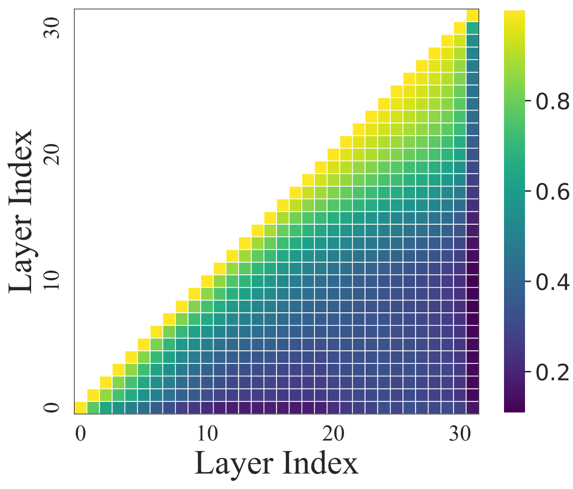


Figure 1: Cosine similarity among layers' representations experimented on LLaMA2-7B model over the GSM8k evaluation dataset.

shown in Figure 1). This effect has also been observed by many prior works (Kornblith et al., 2019; Gromov et al., 2024). This effect is also connected to methods like *layer skipping* and *mixture of depth* for LLMs (Fan et al., 2024; Elhoushi et al., 2024; Raposo et al., 2024).

3 AutoLoRA Method

As illustrated in Figure 2, our method comprises two key components: (i) layer similarity computation and (ii) LoRA sharing.

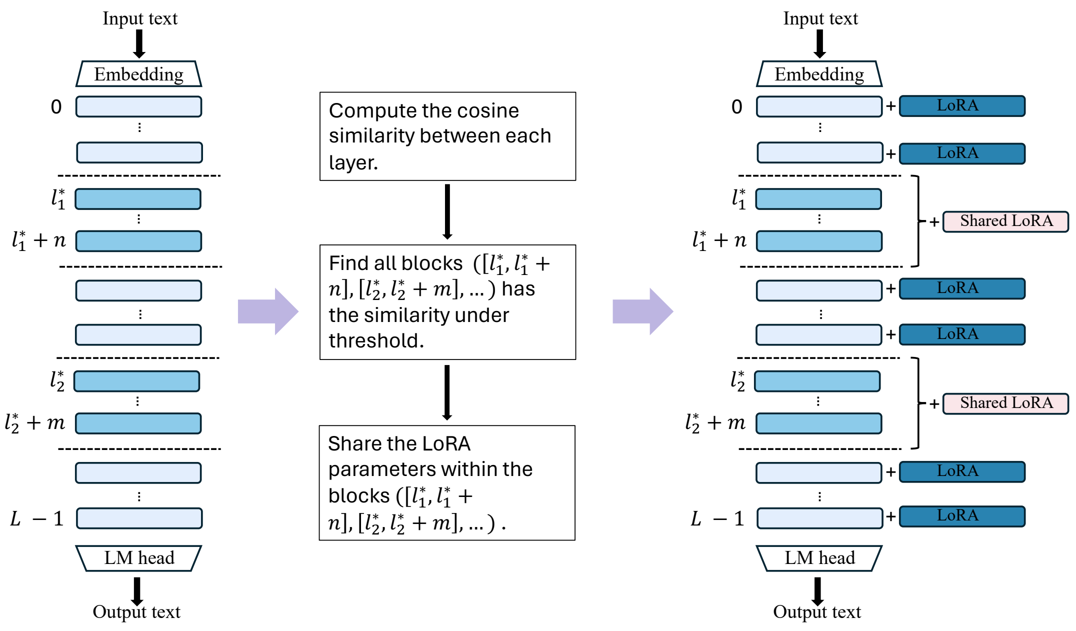
3.1 The Formulation of ShareLoRA

In this subsection, we present the formulation of ShareLoRA as an optimization problem. Specifically, we consider an L-layer LLM. We denote $\mathcal{S}$ as the collection of shared sets of LLM layer indices, *e.g.,* $\mathcal{S} = \{(2, 3), (5, 6, 7), (11, 13)\}$.

Our objective is to maximize the number of shared LLM layers while ensuring that the similarity measure between any two layers with shared weights exceeds a predefined threshold, ϵ. This can be formulated as the following optimization problem:

$$\begin{aligned} \max \quad & \sum_{\forall s \in \mathcal{S}} |s| \\ \text{s.t.} \quad & c(\mathbf{x}_i, \mathbf{x}_i) \geq \epsilon \end{aligned}$$

where $|s|$ is the number of layers in set s, and $\mathcal{S}$ is the collection of sets such that each set s contains layers with similarity measure (*i.e.,* $c(\cdot, \cdot)$) between each pair exceeding the threshold ϵ.



Input text
Embedding
0
l_1^*
$l_1^* + n$
l_2^*
$l_2^* + m$
$L - 1$
LM head
Output text
Compute the cosine similarity between each layer.
Find all blocks ($[l_1^*, l_1^* + n], [l_2^*, l_2^* + m], \ldots$) has the similarity under threshold.
Share the LoRA parameters within the blocks ($[l_1^*, l_1^* + n], [l_2^*, l_2^* + m], \ldots$).
LoRA
Shared LoRA

Figure 2: An overview of our proposed AutoLoRA, which computes the cosine similarity between each layer, and shares the LoRA parameters within redundancy blocks.

The objective of SHARELORA is to maximize the total number of layers sharing LoRA. The remaining question is *How to find $\mathcal{S}$*? We will present our algorithm of SHARELORA in section 3.3.

3.2 Similarity Between Representations of Layers

To decide which layer's LoRA module could be shared with another, we have to compute the angular distance $c(\mathbf{x}_i, \mathbf{x}_j)$ between the representations of layer i and layer j. The similarity of a single sequence of length T is given by

$$c(\mathbf{x}_i, \mathbf{x}_j) := \frac{1}{\pi} \cos^{-1}\left(\frac{\mathbf{x}_i^\top \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}\right), \quad (2)$$

where the inner product is over the hidden dimension of the model for the final token T of the sequence, $\|\cdot\|$ denotes the ℓ_2-norm, and the factor of $1/\pi$ is a convention. This distance should then be summed over a number of examples that is large enough to get a low-fluctuation estimate but overall should be quite small.

3.3 Similarity-based Weight Sharing

The objective of the algorithm is to identify sets of neural network layers that can share LoRA modules based on the angular distance of their representations. The algorithm employs a greedy strategy to maximize the size of these shared sets while adhering to a predefined similarity threshold.

First, the algorithm processes the similarity matrix to create an upper triangular matrix, excluding the diagonal, which represents the pairwise similarities between layers. It then identifies all eligible pairs of layers (i, j) where the similarity score exceeds the threshold ϵ. Formally, this can be expressed as: shared_pairs $= \{(i, j) \mid c(\mathbf{x}_i, \mathbf{x}_j) \geq \epsilon$ and $0 \leq i < j < L\}$.

Next, the algorithm constructs sets of shared layers by iterating through the identified shared pairs. It maintains a set visited to avoid reprocessing layers. The construction of the shared sets proceeds is shown in Algorithm 1.

The algorithm ensures that layers are grouped into shared sets only if their pairwise similarities exceed the threshold ϵ, thus maximizing the number of layers that can share the LoRA modules while maintaining high similarity within each set.

Furthermore, after sharing layers using the set $\mathcal{S}$, the number of layers with shared parameters will be $\sum_{S_i \in \mathcal{S}} |S_i| - |\mathcal{S}|$. This allows us to optionally expand the original rank r to $\lfloor \frac{L}{L - \sum_{S_i \in \mathcal{S}} |S_i| + |\mathcal{S}|} r \rfloor$, where L is the total number of layers. We summarize the algorithm in Algorithm 2.

Algorithm 1 Construct Shared Sets

```
1: Input: Similarity matrix $C$, threshold $\epsilon$;
2: Output: Collection of shared sets $\mathcal{S}$;
3: Initialize shared_pairs $\leftarrow \{(i,j) \mid c(\mathbf{x}_i, \mathbf{x}_j) \geq \epsilon$ and $0 \leq i < j < L\}$ and sort it;
4: Initialize $\mathcal{S} \leftarrow \{\}$;
5: Initialize visited $\leftarrow$ set();
6: for all $(i,j)$ in shared_pairs do
7:   if $i$ not in visited and $j$ not in visited then
8:     if $i$ not in $\mathcal{S}$ then
9:       Initialize $S_i \leftarrow \{i\}$
10:    end if
11:    Add $j$ to $S_i$;
12:    Add $j$ to visited
13:    if All pairs starting with $i$ is done then
14:      Add $i$ to visited
15:    end if
16:  end if
17: end for
```

Algorithm 2 SHARELORA

```
1: Input: Sample Dataset $\mathcal{D}^*$; Train Dataset $\mathcal{D}$; hyper-parameter similarity threshold $\epsilon$.
2: Inference on $\mathcal{D}^*$ and save the representations $x(l)$ for each layer $l$;
3: Compute the angular distance $d(x(l_p), x(l_q))$ between each layer $l_p$ and $l_q$;
4: Compute the share set $S$ and update LoRA rank $r$;
5: Share parameters using $S$;
6: Finetune weight sharing model $W$ with $\mathcal{D}$.
7: Output: The fine-tuned parameters $W^*$.
```

4 Experiments

Models and datasets. We implemented SHARELORA to fine-tune LLaMA family LLMs, namely, LLaMA-7B (Touvron et al., 2023a), LLaMA2-7B (Touvron et al., 2023b) and LLaMA3-8B (Meta, 2024). We follow the settings of LLM-Adapters (Hu et al., 2023), and evaluate the effectiveness of the several natural language commonsense reasoning task including BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2019), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2021), ARC-Easy (Clark et al., 2018), ARC-Challenge, and OBQA (Mihaylov et al., 2018). Additionally, we implemented SHARELORA to fine-tune LLaVA-1.5-7B (Liu et al., 2023), a popular vision language foundation model (VLM) and used on image-text pair understanding, and evaluated on LLaVA-Bench (in-the-wild) evaluation dataset (Liu et al., 2023).

Setup. We use PyTorch (Paszke et al., 2019) to implement all the algorithms. Our fine-tuning algorithm implementation is based on the publicly available Huggingface Transformers (Wolf et al., 2019) and LLM-Adapters code base. All the experiments are conducted on NVIDIA A6000 GPUs.

Baselines. We compare SHARELORAwith the following baselines.

- *Prompt learning (Prefix):* (Li and Liang, 2021) Involves fine-tuning a small set of continuous task-specific vectors (prefixes) while keeping the large language model parameters frozen to pre-trained weights.
- *Adapter tuning (AdapterH):* (Houlsby et al., 2019) Inserts small add-on layers between the multi-head attention modules and FFN modules of the the pre-trained model that can be fine-tuned for downstream task learning, while keeping the rest of the model parameters frozen.
- *Pfeiffer adapter (AdapterP):* (Pfeiffer et al., 2020) Unlike adapter tuning it inserts add-on layers after FFN modules and LayerNorm modules, allows them to fine-tune and keeps the rest of the model frozen.
- *Parallel adapter (Parallel):* (He et al., 2021) Modifies the hidden representations in a transformer model by inserting additional trainable parameters in parallel to the original model's layers.
- *LoRA:* (Hu et al., 2021) Is the most popular PEFT method that injects trainable low-rank matrices into transformer layers parallel to the frozen main path, to approximate the weight updates.

4.1 Evaluations on LLMs

We assess the fine-tuning performance of LLaMA-7B, LLaMA2-7B, and LLaMA3-8B using all baseline methods along with the proposed algorithm. The commonsense reasoning evaluation includes eight sub-tasks, each with its own predefined training and testing sets. Following the setup from LLM-Adapters (Hu et al., 2023), we combine the training datasets from all the eight tasks to form a comprehensive training dataset for the fine-tuning, and

Model	PEFT Method	# Params	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
LLaMA-7B	Prefix	10.5M	57.46	50.49	33.62	28.38	64.8	29.84	24.49	26.6	39.46
	AdapterH	20.1M	71.19	74.48	45.45	57.24	59.51	56.90	33.70	39.0	54.68
	AdapaterP	20.1M	67.65	73.45	44.27	57.04	58.48	57.62	33.87	37.0	53.67
	Parallel	448M	73.36	74.54	**73.81**	57.08	60.22	56.23	35.58	36.8	54.70
	LoRA($r = 8$)	14.0M	**78.53**	78.45	46.98	**73.41**	70.17	**70.24**	**41.21**	42.2	62.65
	ShareLoRA(ours)	**10.8M**	77.55	**79.65**	47.54	73.04	**70.96**	69.95	40.70	**45.4**	**63.10**
LLaMA2-7B	LoRA($r = 8$)	14.0M	80.64	79.16	**47.85**	**75.18**	69.93	**69.70**	**42.06**	**44.0**	**63.56**
	ShareLoRA(ours)	**11.8 M**	**80.89**	**79.22**	46.78	74.04	**71.27**	68.73	41.81	**44.0**	63.36
LLaMA3-8B	LoRA($r = 8$)	14.2M	**82.17**	81.34	**49.49**	78.38	**74.19**	76.39	**50.77**	**46.4**	**67.39**
	ShareLoRA(ours)	**11.4M**	**82.17**	**81.56**	48.77	**79.32**	73.80	**76.85**	50.00	44.2	67.07

Table 2: Accuracy with LLaMA model family with various PEFT methods on commonsense reasoning tasks.

perform evaluations on the individual testing sets for each sub-task.

Implementation details. All of LLaMA models have 32 hidden layers. Initially, we calculate the similarity $c(\mathbf{x}_i, \mathbf{x}_i)$ between each layer's representation using 256 randomly sampled C4 validation dataset. We set the similarity threshold ϵ to 0.85 for LLaMA-7B and LLaMA2-7B, and 0.80 for LLaMA3-8B. Utilizing our similarity-based weight-sharing algorithm, the share set collection $\mathcal{S}$ for LLaMA-7B is $\{\{16, 17\}, \{18, 19\}, \{20, 21, 22\}, \{23, 24, 25, 26, 27\}, \{28, 29, 30\}\}$, enabling the sharing the LoRA modules of **ten** similar layers. The share set collection $\mathcal{S}$ for LLaMA2-7B and LLaMA3-8B are $\{\{17, 18\}, \{19, 20\}, \{21, 22, 23\}, \{24, 25, 26, 27\}, \{28, 29\}\}$ and $\{\{19, 20\}, \{21, 22\}, \{23, 24, 25\}, \{26, 27, 28\}\}$ separate. Note, for these three models, we expand the original LoRA rank from 8 to 9.

Table 2 shows experimental results on the eight commonsense reasoning tasks. ShareLoRA achieves similar or better performance across all datasets for all the models. Notably, our proposed approach saves up to **23**% of the trainable parameters yet achieves a **1.5**% improvement in performance with LLaMA-7B, while maintaining similar performance on LLaMA2-7B and LLaMA3-8B.

4.2 Evaluations on Multi-Modal VLMs

We now present the results of ShareLoRAon vision-language models. We used the LLaVA-1.5-7B (Liu et al., 2023), which consists of a language model, a visual encoder and a projection layer for feature alignment. Specifically, the language model and visual encoder were initialized with Vicuna-1.5-7B (Zheng et al., 2024) and CLIP ViT-L/336px (Radford et al., 2021), respectively. In addition, we employed the pre-trained projection layer[1] and directly performed visual instruction fine-tuning stage. In contrast to the setup of LLaVA, we chose a subset of their dataset, i. e. LLaVA-Instruct-80K. It consists of 80k image-instruction pairs filtered out from LLaVA-Instruct-150K[2]. We set the rank to 64 and performed instruction tuning for one epoch of the LLM component using ShareLoRAand LoRA, respectively. Note, we keep the feature transformation module trainable for both.

Implementation details. For ShareLoRA we compute the similarity between the inputs and outputs of each layer based on 128 samples, with each sample comprising of visual-text inputs. The similarity threshold was set to 0.9 and the share set collection S for language model is $\{\{2, 3\}, \{4, 5\}, \{6, 7\}, \{17, 18\}, \{19, 20\}, \{21, 22, 23\}, \{24, 25, 26\}, \{27, 28, 29\}\}$. Please see Table 3 for detailed hyperparameters setting. We use GPT-4 generated answers as the golden answers. Subsequently, we employed GPT-4o to evaluate the instruction-following capabilities of the fine-tuned model and selected the challenging LLaVA-Bench (in-the-wild). It comprises 24 images with a total of 60 questions. In Table 4, we demonstrate the performance of ShareLoRA and LoRA fine-tuned LLaVA model. Overall, ShareLoRA exhibits superior performance compared to that with LoRA, with a significant advantage in the conversation aspect. Additionally, ShareLoRArequires around **13**% fewer trainable parameters to yield this improved performance.

4.3 Ablations and Discussions

Robustness towards different rank setting. Here we investigate the impact of various rank configurations on ShareLoRA and LoRA by adjusting the original r within the set $\{2, 4, 8, 16\}$. We then evaluate the performance of fine-tuned LLaMA-7B on commonsense reasoning tasks as described in §4.1. Fig. 3 depicts the results with different ranks for both LoRA and ShareLoRA. Across

[1]https://huggingface.co/liuhaotian/llava-v1.5-mlp2x-336px-pretrain-vicuna-7b-v1.5

[2]https://huggingface.co/datasets/liuhaotian/LLaVA-Instruct-150K

Hyperparameters	LoRA	ShareLoRA
Similarity threshold	—	0.9
Rank r	64	64
α	128	128
Dropout	0.05	0.05
Target layer	Q,K,V,O,Up,Down,Gate	Q,K,V,O,Up,Down,Gate
Epochs	1	1
LR	2e-4	2e-4
LR Scheduler	Cosine	Cosine
Optimizer	AdamW	AdamW
Batch Size	64	64
Warmup Ratio	0.03	0.03

Table 3: Hyperparameters of LoRA and ShareLoRA for fine-tuning vision-language model.

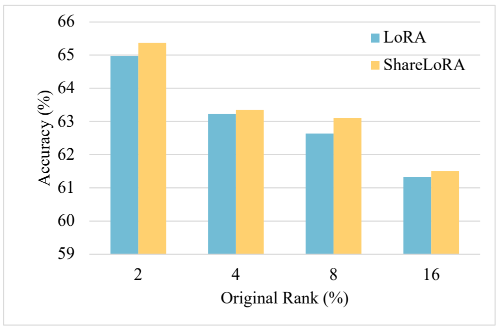


Figure 3: Accuracy comparison with different rank values for LoRA and ShareLoRA.

	LoRA(rank=64)	ShareLoRA
# Params	159.9M	**139.4M**
Description	45.57 $\pm$ 0.60	**45.80** $\pm$ 0.40
Conversation	46.87 $\pm$ 1.27	**54.10** $\pm$ 0.00
Reasoning	**66.90** $\pm$ 0.85	65.37 $\pm$ 0.72
All	55.47 $\pm$ 0.55	**57.00** $\pm$ 0.44

Table 4: Instruction-following capability comparison between LoRA and ShareLoRA (ours). We conduct three repeated evaluations and report the average scores. The results are reported in the format of $mean \pm std$.

Target Modules	Accuracy
Q	64.19
K	64.31
V	63.71
Down	62.63
Up	62.91
Q,K,V,Down,Up	63.10

Table 6: Accuracy comparison of placing LoRA at different target modules.

all four original rank settings, ShareLoRA consistently improves performance compared to the baseline LoRA. Despite using the same original rank, ShareLoRA employs only 80% of the trainable parameters relative to the LoRA. For instance, ShareLoRA achieves an average accuracy of 63.35% on the eight commonsense reasoning tasks with 5M trainable parameters, whereas LoRA requires 7M parameters for the same rank level.

Similarity threshold. Table 5 presents the experimental results of fine-tuning LLaMA-7B with different similarity thresholds $\{0.75, 0.80, 0.85, 0.90\}$. The best performance is achieved when the similarity threshold ϵ is set to 0.85. Higher similarity thresholds result in more eligible shared layers. As the number of shared layers increases, the rank of the LoRA modules can be adjusted upward within the constraints of the trainable parameter budget, which is determined by the original rank. Thus, there is a tradeoff between having more independent layers and a larger rank. Consequently, selecting an appropriate similarity threshold ϵ is crucial for optimizing the performance of our proposed method.

Similarity threshold	Accuracy
0.75	61.80
0.80	62.81
0.85	63.10
0.90	61.80

Table 5: Accuracy comparison of different similarity thresholds evaluated with LLaMA-7B.

Adapter sensitivity to different layer types. In subsection 4.1, we adapt our proposed method to the Q, K, V, Down, and Up weights. Table 6 presents the performance of fine-tuning the weights associated to each module type separately. We use $r = 8$ for the LoRA modules. The results indicate that adapting our proposed method to the Q and K weights yields the most significant benefits in fine-tuning. This improvement can be attributed to the critical role these weights play in attention mechanisms, highlighting the importance of carefully selecting target modules for optimization.

5 Related Works

Generative foundation models. Generative deep learning models pre-trained on large datasets are called generative foundation models (Bommasani et al., 2021). These foundation models can be applied to downstream tasks by fine-tuning. Advanced generative foundation models in natural language processing (NLP) such as GPT (Brown et al., 2020; Ouyang et al., 2022) and LLaMA (Touvron et al., 2023a) model have shown great success in assisting and generating human-like text across a wide range of topics. These generative language models can also be applied to many practical downstream tasks, such as education (Kasneci et al., 2023) and healthcare (Thirunavukarasu et al., 2023). Another kind of generative foundation model that has developed maturely is the diffusion model (Ho et al., 2020; Rombach et al., 2022). The diffusion model works well in various computer vision tasks such as text-to-image generation (Everaert et al., 2023) and image editing (Kawar et al., 2023).

Efficient fine-tuning methods. Efficient fine-tuning methods aim to reduce the number of trainable parameters to save the GPU memory and training time during fine-tuning large-scale models. Some PEFT methods freeze most of the parameters in the model and only fine-tune specific modules, e.g., BitFit (Zaken et al., 2021) fine-tunes only the bias of the model, which significantly saves the GPU memory. However, it cannot be executed on models without bias parameters. (Houlsby et al., 2019) and (Pfeiffer et al., 2020) add adapter layers between transformer blocks. These methods accelerate fine-tuning by transferring knowledge from adapter layers pre-trained on general tasks. LoRA (Hu et al., 2021; Liu et al., 2024b) is the most popular adapter for fine-tuning large foundation models. It adopts the product of two small matrices to represent the full gradient during fine-tuning. It can reduce the number of trainable parameters by 10,000 times and the GPU memory requirement by 3 times. Some adaptive algorithms work together with LoRA that can dynamically adjust the number of trainable parameters to fit specific needs. For instance, AdaLoRA (Zhang et al., 2023) adaptively allocates the trainable parameters to fit the GPU memory budget. These adaptive algorithms on LoRA require heterogeneous LoRA configuration when implemented in federated fine-tuning.

6 Future work

Our proposed SHARELORA method is orthogonal to other LoRA variants, suggesting that future work could explore combining SHARELORA with existing PEFT methods. For example, LoRA+ (Hayou et al., 2024) sets different learning rates for the adapter matrices A and B. By adjusting the learning rate ratio between these two matrices, the efficiency and performance of fine-tuning can be significantly improved. Specifically, LoRA+ sets the learning rate of B to be λ times that of A. This technique could easily be integrated with SHARELORA. Additionally, DoRA (Liu et al., 2024a) is a novel PEFT method that enhances LoRA by decomposing pre-trained weights into magnitude and direction components for more precise fine-tuning. This weight decomposition allows DoRA to optimize the magnitude and direction of weights separately. SHARELORA could also be combined with DoRA to share the magnitude and direction components among redundant layers. In summary, SHARELORA not only offers a standalone solution for fine-tuning but also provides a flexible framework that can be enhanced by integrating with other advanced PEFT methods. This opens up new avenues for research and development, potentially leading to even greater improvements in PEFT performance and efficiency.

7 Conclusion

In this paper, we present SHARELORA, a parameter-efficient fine-tuning method that determines which layers share the LoRA weights based on layer redundancy. SHARELORA leverages the cosine similarity between each layer's representations to ascertain redundancy. Utilizing a greedy algorithm, we maximize the sharing of LoRA weights while adhering to a predefined similarity threshold. This approach effectively reduces the number of trainable parameters. We conduct extensive experiments on large language models and multi-modal vision-language foundation models. The results demonstrate that SHARELORA achieves comparable or superior performance to existing PEFT methods, while using only 80% of the trainable parameter budget.

8 Limitations

There are two limitations to this work. Firstly, due to constrained computing resources, we were unable to evaluate the performance of larger language models such as LLaMA2-70B and LLaMA3-70B. It is anticipated that these larger models, containing more redundancy with a fixed number of training datasets, would yield superior performance. Secondly, this paper does not explore other, more precise metrics for identifying layer redundancy in large language models. Designing more precise and fine-grind metrics to determine which layer is redundancy is still a challenge, we intend to investigate this direction in future research endeavors.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

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

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

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

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. *Preprint*, arXiv:1905.10044.

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. *arXiv preprint arXiv:1803.05457*.

Dawei Dai, Liping Yu, and Hui Wei. 2020. Parameters sharing in residual neural networks. *Neural Processing Letters*, 51(2):1393–1410.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Søgaard. 2018. Parameter sharing between dependency parsers for related languages. *arXiv preprint arXiv:1808.09055*.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. 2024. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*.

Martin Nicolas Everaert, Marco Bocchio, Sami Arpa, Sabine Süsstrunk, and Radhakrishna Achanta. 2023. Diffusion in style. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2251–2261.

Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, Yequan Wang, and Zhongyuan Wang. 2024. Not all layers of llms are necessary during inference. *arXiv preprint arXiv:2403.02181*.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. 2024. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*.

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 (Volume 1: Long Papers)*, pages 4884–4896.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

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 *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.

Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274.

Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. 2023. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6007–6017.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. 2018. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024a. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Zeyu Liu, Souvik Kundu, Anni Li, Junrui Wan, Lianghao Jiang, and Peter Anthony Beerel. 2024b. Aflora: Adaptive freezing of low rank adaptation in parameter efficient fine-tuning of large models. *arXiv preprint arXiv:2403.13269*.

AI Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*.

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. *arXiv preprint arXiv:1809.02789*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. 2024. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2022. Large language models encode clinical knowledge. *arXiv preprint arXiv:2212.13138*.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, et al. 2023. Towards expert-level medical question answering with large language models. *arXiv preprint arXiv:2305.09617*.

Sho Takase and Shun Kiyono. 2021. Lessons on parameter sharing across layers in transformers. *arXiv preprint arXiv:2104.06022*.

MosaicML NLP Team. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. Accessed: 2023-05-05.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature medicine*, 29(8):1930–1940.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

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*.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.