ShareLoRA: Parameter Efficient and Robust Large Language Model Fine-tuning via Shared Low-Rank Adaptation

Anonymous ACL submission

Abstract

In this paper, we introduce **Share**d **Lo**w **R**ank **A**daptation (ShareLoRA) for parameter efficient and robust Large Language Model (LLM) fine-tuning. By strategically deploying ShareLoRA across different layers and adapting it for the Query, Key, and Value components of self-attention layers, we achieve a substantial reduction in the number of training parameters and memory usage. Importantly, ShareLoRA not only maintains model performance but also exhibits robustness in both classification and generation tasks across a variety of models, including RoBERTa, GPT-2, and LLaMA series. It demonstrates superior transfer learning capabilities compared to standard LoRA applications and mitigates overfitting by sharing weights across layers. Our findings affirm that ShareLoRA effectively boosts parameter efficiency while ensuring scalable and high-quality performance across different language model architectures.

1 Introduction

As Pretrained Language Models (PLMs) have gained prominence (Devlin et al., 2019; Liu et al., 2019; Radford et al., 2019), researchers are increasingly focused on optimizing the utilization of these models' pre-trained weights. Traditional fine-tuning, which involves adjusting all parameters of a PLM for a specific dataset or task, is often resource-intensive and time-consuming, especially given the massive scale of large language models (LLMs) (Brown and et.al, 2020; Kaplan et al., 2020; Hoffmann and et.al, 2022; et.al, 2022; Zhang et al., 2022; et.al, 2023b).

Parameter-Efficient Fine-Tuning (PEFT) has proven to be an effective strategy for mitigating the challenges associated with extensive parameter adjustments. By modifying only a select subset of a model's parameters, PEFT enables cost-effective adaptation to domain-specific tasks while preserving performance levels comparable to those achieved with full fine-tuning (Houlsby et al., 2019; Li and Liang, 2021a; Lin et al., 2020; Lei et al., 2023; He et al., 2022, 2023; Mahabadi et al., 2021). Techniques like Low-Rank Adaptation (LoRA) (Hu et al., 2021) stand out within PEFT by demonstrating that models fine-tuned with a reduced parameter set can match the performance of those fine-tuned with full parameters, effectively bridging the gap in efficiency and efficacy.

Given the impressive performance of LoRA, subsequent studies have aimed to enhance its efficiency, mainly by reducing the number of trainable parameters to minimize the memory footprint during the fine-tuning process. However, significantly lowering the trainable parameters can lead to slow convergence, while insufficient reductions may encourage the model to easily overfit. Moreover, existing PEFT methods often struggle to maintain robustness across different domains after fine-tuning.

To address these challenges, we introduce ShareLoRA, an efficient and straightforward PEFT method that effectively balances trainable parameter selection while optimizing the model's adaptability, minimizing memory requirements, and ensuring robustness across domains. Our approach leverages the observation that low-rank weight matrices A and B do not need to be uniquely configured across layers to achieve optimal PEFT performance in PLMs. Instead, we propose sharing either matrix A or B across all layers while maintaining its counterpart as distinct in each layer. This strategy meets several key objectives:

1. Parameter Efficiency: Sharing a low-rank matrix across layers significantly reduces the number of trainable parameters and cuts down on the memory footprint needed for model fine-tuning. Our experiments demonstrate that ShareLoRA achieves comparable or superior performance to other PEFT methods while using fewer parameters, making it one of the most parameter-efficient approaches available.

2. Model Adaptability: Keeping the shared matrix trainable preserves the model's adaptability, allowing it to effectively learn and adapt to new tasks and domains. Also, the updated weights for each component that LoRA applies remain unique yet share a common base, promoting consistency across layers while allowing for task-specific adaptations.

3. Cross-Domain Robustness: ShareLoRA exhibits robustness when applied to domains different from the one it was fine-tuned on. This cross-domain generalization capability sets it apart from traditional LoRA and other PEFT methods, which often struggle to maintain performance when faced with out-of-domain tasks.

Our extensive experiments across multiple models, including RoBERTa, GPT-2, and LLaMA series, demonstrate that ShareLoRA not only preserves model performance but also shows remarkable robustness across a variety of tasks in both classification and generation. ShareLoRA significantly improves parameter efficiency while maintaining scalable and high-quality performance across diverse language model architectures. Moreover, it exhibits enhanced transfer learning capabilities and robust performance on out-of-domain tasks, effectively addressing the challenges of overfitting often associated with traditional fine-tuning methods.

2 Related Works

PLMs are trained on large datasets to develop broad linguistic representations (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2020), but often fall short in specialized tasks due to a lack of domain knowledge. Traditional approaches involve fully fine-tuning PLMs to enhance domain-specific performance (Xu and Wang, 2023; Xie et al., 2020; Dabre et al., 2019). However, with the increasing size of PLMs (Workshop et al., 2023; et.al, 2023b,a; Zhang et al., 2022), this method becomes too resource-heavy. As an alternative, Parameter Efficient Fine-tuning (PEFT) provides an efficient way to maintain performance with less computational expense.

PEFT methods have become crucial for adapting large-scale pre-trained models to specific tasks without extensively overhauling their parameters. This approach conserves computational resources and boosts efficiency. For example, Prefix tuning (Li and Liang, 2021a) adds parameters to the hidden states across layers, subtly influencing the model's behavior without changing its underlying architecture. Prompt tuning (Lester et al., 2021) alters prompts and updates only the associated parameters, focusing on specific areas of model performance. BitFit (Zaken et al., 2022) updates only the biases within the model, resulting in minimal yet effective modifications.

One notable PEFT technique is Low-Rank Adaptation (LoRA) (Hu et al., 2021), which achieves efficient fine-tuning by incorporating a low-rank matrix adaptation mechanism alongside the existing weights of linear layers. This approach reduces memory overhead while preserving the effectiveness of the fine-tuning process. The modified output Y is computed as follows:

$$Y \leftarrow XW + \alpha XAB \tag{1}$$

where W represents the original pre-trained weights, X is the input tensor, and α is a scaling factor. The adaptation is facilitated by matrices A and B, where $A \in \mathbb{R}^{d_{\text{in}} \times r}$ projects the input dimension down to a lower rank r, and $B \in \mathbb{R}^{r \times d_{\text{out}}}$ projects it back up, effectively creating a bottleneck that captures the most significant transformations.

Recent enhancements to LoRA have significantly broadened its capabilities. QLoRA (Dettmers et al., 2023) optimizes LoRA for the fine-tuning of quantized models, thereby increasing efficiency. ReLoRA (Lialin et al., 2023) incorporates a warm-up strategy during pre-training to boost adaptability. LoraHub (Huang et al., 2024) streamlines the process by automating the creation of custom LoRA modules for specific tasks. Additionally, GLoRA (Chavan et al., 2023) introduces a prompt module that fine-tunes weights and biases, enhancing performance across a variety of applications.

Despite these advancements, LoRA still faces significant memory overhead due to high activation memory usage in LoRA layers during the fine-tuning phase. To address this issue, LoRA-FA (Zhang et al., 2023) strategically freezes the low-rank A matrix and updates only the B matrix. This approach significantly reduces the number of trainable parameters and activation memory, thus enhancing the efficiency of fine-tuning large language models without substantially impacting performance.

However, LoRA-FA does not adequately decrease the total number of parameters that need to be stored, presenting a considerable challenge in

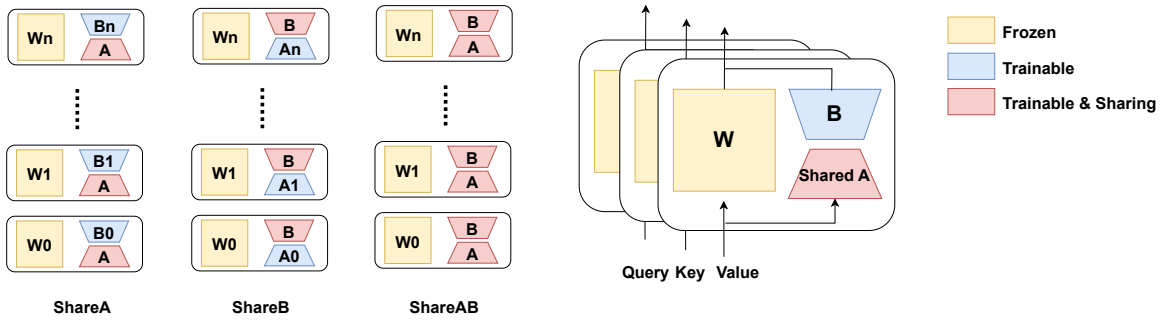

Figure 1: Overview of ShareLoRA: The implementation of ShareA, ShareB, and ShareAB across all layers (left), including ShareA applied across self-attention layers (right).

contexts where computational resources and storage are constrained. Additionally, by freezing the A matrix, LoRA-FA limits the model's capacity to adapt and learn from new data during fine-tuning. This rigidity can hinder the model's performance, particularly in complex or domain-specific tasks.

In contrast, our proposed approach ShareLoRA offers a more dynamic and flexible strategy by allowing either matrix A or B, or both, to be shared across different layers. This method not only preserves the model's adaptability but also further reduces the memory requirements. The details of ShareLoRA will be elaborated in the following sections.

3 Method

In this section, we provide a detailed description of our proposed PEFT approach ShareLoRA, as illustrated in Figure1. ShareLoRA facilitates flexible configurations through two primary dimensions: **1.** the choice of sharing between the matrices A, B, or both A and B (ShareA, ShareB, and ShareAB), and **2.** the scope of sharing, which can be across different layers such as self-attention layers. This framework allows for a variety of combinations, enabling tailored adaptation of low-rank models to specific tasks.

ShareA Configuration In the ShareA configuration, the low-rank matrix A is uniformly shared across all layers, with each layer employing its own unique matrix B_i. The formula for weight adaptation in each layer i can be expanded to detail the influence on model transformation:

$$\Delta W_i = \alpha A B_i = \alpha \sum_{k=1}^{r} A_{:,k} B_{k,:,i} \tag{2}$$

where $A_{:,k}$ represents the k-th column of A, and $B_{k,:,i}$ is the k-th row of matrix B_i. This equation shows that each layer's weight change, ΔW_i, is a linear combination of the columns of A weighted by the corresponding elements of B_i. This shared projection-down matrix A reduces the dimensionality uniformly across all layers, thereby minimizing redundancy in learning and memory usage while enabling tailored output transformations through layer-specific matrices B_i.

ShareB Configuration In the ShareB configuration, matrix B is uniformly shared across all layers, while each layer employs its own unique matrix A_i. The weight adjustment for each layer is expressed as:

$$\Delta W_i = \alpha A_i B = \alpha \sum_{k=1}^{r} A_{i,:,k} B_{k,:} \tag{3}$$

where $A_{i,:,k}$ denotes the k-th column of matrix A_i for layer i, and $B_{k,:}$ represents the k-th row of the shared matrix B. Here, the uniform projection-up matrix B ensures consistent expansion of the transformed data back to the output dimension across all layers, while the distinct A_i matrices allow for adaptation to the specific input characteristics of each layer.

ShareAB Configuration When both matrices A and B are shared across all layers, the change in weights is simplified, leading to substantial parameter reduction:

$$\Delta W = \alpha A B = \alpha \sum_{k=1}^{r} A_{:,k} B_{k,:} \tag{4}$$

where both $A_{:,k}$ and $B_{k,:}$ are shared across all layers. This configuration significantly reduces the model complexity by eliminating the need for distinct matrices in each layer, thus reducing memory

requirements and computational overhead. The entire model operates under a uniform transformation schema, which simplifies training and storage but requires careful calibration of the initial values and ongoing adjustments during fine-tuning to preserve model effectiveness across diverse tasks.

Sharing Across Self-Attention Layers In the ShareA configuration of ShareLoRA applied to PLMs across all self-attention layers, the matrices A_Q, A_K, and A_V are shared. These matrices are responsible for reducing the dimensionality of the inputs for Queries (Q), Keys (K), and Values (V) respectively, we term it as **ShareA**$_{qkv}$ in the following paragraphs. The process for each component in the i-th self-attention layer is formalized as follows:

$$Q_i = X_i A_Q B_{Q_i} \tag{5}$$

$$K_i = X_i A_K B_{K_i} \tag{6}$$

$$V_i = X_i A_V B_{V_i} \tag{7}$$

$$\text{Attention}(Q_i, K_i, V_i) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_{K_i}}}\right) V_i, \tag{8}$$

where X_i denotes the input to the i-th self-attention layer. Each matrix A_Q, A_K, and A_V facilitates a consistent reduction in input dimensions across all layers, which simplifies the model architecture by maintaining a uniform approach to processing the foundational aspects of self-attention. The unique matrices B_{Q_i}, B_{K_i}, and B_{V_i} for each component allow for tailored transformations that meet the specific needs of each self-attention layer.

4 Experiments

In our study, we conduct a comprehensive evaluation of the downstream performance of ShareLoRA across several series models, including RoBERTa (Liu et al., 2019) and GPT-2 (Radford et al., 2019). We benchmark these results against other established approaches such as LoRA (Hu et al., 2021), LoRA-FA (Zhang et al., 2023). Additionally, we extend the application of ShareLoRA to large-scale model in LLaMA series (et.al, 2023b, et.al, 2023a, Dubey et al., 2024) architectures, particularly in few-shot, zero-shot scenarios. Furthermore, our experiments cover a range of model sizes, from 7 billion to 13 billion parameters, and included both quantized and unquantized model variants. All tests were performed on the Nvidia A6000 and RTX 3090 GPUs. For experiment hyper-parameter settings, see Appendix Table 7-Table 10.

4.1 Datasets

The experiment datasets are primarily divided into three categories: Natural Language Understanding (NLU), Natural Language Generation (NLG) and few-shot tasks, using the same configuration and datasets as LoRA (Hu et al., 2021) and (Dettmers et al., 2023).

For NLU, we employ the GLUE benchmark (Wang et al., 2019), which includes MNLI, SST-2, MRPC, CoLA, QNLI, QQP, RTE, and STS-B tasks. Notably, for MRPC, RTE, and STS-B tasks, we initialize the LoRA modules with the trained MNLI checkpoint as (Hu et al., 2021) demonstrated. For NLG, we replicate experiments similar to those of LoRA using the E2E challenge dataset (Novikova et al., 2017), following the same experimental setup.

Additionally, we expand our experiments to few-shot and zero-shot tasks on larger models, demonstrating our approach's adaptability. Following the configuration outlined in (Dettmers et al., 2023), we employ Alpaca (Taori et al., 2023) and MATH (Hendrycks et al., 2021b) for LoRA and ShareLoRA, using the MMLU benchmark (Hendrycks et al., 2021a) for evaluation. Some other benchmarks like ARC (Chollet, 2019), Hellaswrag (Zellers et al., 2019), MMLU-Pro (Wang et al., 2024) and GSM8K (Cobbe et al., 2021) are used for comparison of model adaptability. All experimental setups are consistent with those described studies and demonstration of their repositories, based on the best of our knowledge.

4.2 Baselines

Full Fine-Tuning (FT) is a commonly used approach for model adaptation involving with updating all model's parameters.

LoRA (Hu et al., 2021) is a technique that introduces a pair of rank decomposition trainable matrices alongside existing weight matrices in neural networks.

Bitfit is a technique studied by (Zaken et al., 2022) for updating only a select small subset of biases parameters, to improve performance on new tasks while freezing all other pre-trained weights.

PreLayer/Prefix (Li and Liang, 2021b) is a parameter-efficient technique for customizing large language models by learning specific activations after each Transformer layer for designated prefix

Method	# Params	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
R_b (FT)*	125.0M	**87.6**	94.8	90.2	63.6	92.8	**91.9**	78.7	91.2	86.4
R_b (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
R_b ($Adpt^D$)*	0.3M	$87.1_{\pm.0}$	$94.2_{\pm.1}$	$88.5_{\pm1.1}$	$60.8_{\pm.4}$	$93.1_{\pm.1}$	$90.2_{\pm00}$	$71.5_{\pm2.7}$	$89.7_{\pm.3}$	84.4
R_b ($Adpt^D$)*	0.9M	$87.3_{\pm.1}$	$94.7_{\pm.3}$	$88.4_{\pm.1}$	$62.6_{\pm.9}$	$93.0_{\pm.2}$	$90.6_{\pm.0}$	$75.9_{\pm2.2}$	$90.3_{\pm.1}$	85.4
R_b (Prefix)*	0.36M	85.21	93.81	87.25	59.31	90.77	87.75	54.51	88.48	80.9
R_b (IA^3)*	0.06M	83.95	93.92	87.00	59.58	90.88	87.99	71.12	90.30	83.1
R_b (LoRA)*	0.3M	$87.5_{\pm.3}$	$\mathbf{95.1_{\pm.2}}$	$89.7_{\pm.7}$	$63.4_{\pm1.2}$	$\mathbf{93.3_{\pm.3}}$	$90.8_{\pm.1}$	$86.6_{\pm.7}$	$\mathbf{91.5_{\pm.2}}$	**87.2**
R_b (L-FA)*	0.15M	86.8	94.8	90	63.6	92.5	90.1	67.9	89.6	84.4
R_b (ShareA)	0.16M	$87.3_{\pm.2}$	$95.0_{\pm.3}$	$89.9_{\pm.8}$	$\mathbf{63.8_{\pm1.1}}$	$92.8_{\pm.18}$	$90.3_{\pm.05}$	$\mathbf{87.1_{\pm.5}}$	$91.4_{\pm.1}$	**87.2**
R_l (FT)*	335.0M	90.2	**96.4**	90.9	68.0	94.7	**92.2**	86.6	92.4	88.9
R_l (LoRA)*	0.8M	$90.6_{\pm.2}$	$96.2_{\pm.5}$	$90.9_{\pm1.2}$	$\mathbf{68.2_{\pm1.9}}$	$94.9_{\pm.3}$	$91.6_{\pm.1}$	$87.4_{\pm1.1}$	$\mathbf{92.6_{\pm.2}}$	89.0
R_l (L-FA)*	0.4M	90.1	96	90	68	94.4	91.1	86.1	92	88.5
R_l (ShareA)	0.4M	$\mathbf{90.7_{\pm.1}}$	$96.1_{\pm.1}$	$\mathbf{91.1_{\pm.8}}$	$67.7_{\pm1.5}$	$\mathbf{95.1_{\pm.1}}$	$91.3_{\pm.1}$	$\mathbf{90.3_{\pm.3}}$	$92.5_{\pm.1}$	**89.3**
R_l (Prefix)*	0.9M	89.30	95.76	88.24	59.01	93.32	88.88	74.01	90.92	84.9
R_l (IA^3)*	0.18M	88.63	94.61	86.52	61.15	94.25	89.45	81.23	92.22	86.0
R_l (LoRA)†	0.8M	$90.6_{\pm.2}$	$\mathbf{96.2_{\pm.5}}$	$90.2_{\pm1.0}$	$\mathbf{68.2_{\pm1.9}}$	$94.8_{\pm.3}$	$\mathbf{91.6_{\pm.2}}$	$85.2_{\pm1.1}$	$\mathbf{92.3_{\pm.5}}$	**88.6**
R_l (ShareAB)†	0.03M	$90.2_{\pm.1}$	$95.9_{\pm.3}$	$89.7_{\pm1.0}$	$62.3_{\pm.9}$	$94.6_{\pm.1}$	$89.7_{\pm.1}$	$83.0_{\pm0.8}$	$90.3_{\pm.2}$	87.0
R_l (ShareB)†	0.4M	$90.4_{\pm.1}$	$96.0_{\pm.3}$	$\mathbf{90.4_{\pm.4}}$	$65.8_{\pm.8}$	$94.6_{\pm.1}$	$91.0_{\pm.1}$	$84.1_{\pm1.2}$	$91.4_{\pm.2}$	88.0
R_l (ShareA)†	0.4M	$\mathbf{90.7_{\pm.1}}$	$96.1_{\pm.1}$	$90.0_{\pm.5}$	$67.7_{\pm1.5}$	$\mathbf{95.0_{\pm.1}}$	$91.3_{\pm.1}$	$\mathbf{85.9_{\pm.8}}$	$91.8_{\pm.2}$	**88.6**

Table 1: RoBERTa_{base} and RoBERTa_{large} with different adaptation methods on the GLUE benchmark. $*$ indicates numbers published in prior works. † indicates runs configured in a setup similar to (Houlsby et al., 2019) and (Hu et al., 2021) for a fair comparison.

tokens, while the main model parameters remain unchanged.

Adapter as introduced by (Houlsby et al., 2019), involves inserting adapter layers between neural modules such as the self-attention and MLP modules, enhancing model flexibility without extensive modifications. AdapterL (Lin et al., 2020) introduce adapters only after the MLP module followed by a LayerNorm, with AdapterD (Rücklé et al., 2021) increases efficiency by omitting some adapter layers.

IA^3 (Liu et al., 2022) is a PEFT approach that enhances model performance by scaling activations with learned vectors.

LoRA-FA (Zhang et al., 2023) is a memory-efficient approach to fine-tuning large language models by reducing the activation memory required.

QLoRA (Dettmers et al., 2023) utilizes a frozen, 4-bit quantized pretrained model and LoRA for efficient gradient propagation.

5 Results

Parameter Efficiency and Performance ShareLoRA demonstrates significant parameter efficiency while maintaining or improving performance across various model sizes and tasks. For large-scale LLaMA models, as shown in Table 2, ShareA reduces trainable parameters by 44% compared to LoRA. Despite this substantial reduction, ShareA achieves comparable or improved MMLU scores, with LLaMA 13B showing an increase from 47.60 to 48.15.

In the context of smaller models, such as GPT-2 on the E2E NLG Challenge, Table 3 shows ShareA exhibits even more impressive efficiency. For GPT-2 Medium, ShareA reduces parameters by 43% while improving performance across all metrics. Specifically, ShareA achieves a BLEU score of 69.7 compared to LoRA's 69.5, and a ROUGE-L score of 71.63 versus 71.51. Similarly, for GPT-2 Large, ShareA halves the parameter count while maintaining or slightly enhancing performance, with BLEU and ROUGE-L scores of 70.0 and 71.74, respectively, compared to LoRA's 69.8 and 71.71.

Notably, while ShareA consistently outperforms LoRA, our experiments show that ShareB and ShareAB generally underperform compared to ShareA. For instance, in the GPT-2 Large model, Table 3 shows ShareB achieves a BLEU score of 69.7 and a ROUGE-L score of 70.94, which are lower than both LoRA and ShareA.

Comparing ShareLoRA to other state-of-the-art PEFT methods, we observe competitive or superior performance. For instance, on the GLUE benchmark using RoBERTa-large, Table 1 shows ShareA achieves an average score of 88.6 on GLUE, compared to 84.9 for Prefix-tuning while using significantly fewer parameters. Even in its most aggressive configuration, ShareAB, with only 0.03M trainable parameters, outperforms IA^3 which uses 0.18M parameters, achieving an aver-

Method	# Params	MMLU	Method	# Params	MMLU
LLaMA 7B *	6738.4M	35.1	LLaMA 13B *	13015M	46.9
LLaMA 7B (LoRA)*	159.9M	40.67	LLaMA 13B (LoRA)*	250.3M	47.49
LLaMA 7B (LoRA)	159.9M	$\mathbf{41.65}_{\pm \mathbf{1.0}}$	LLaMA 13B (LoRA)	250.3M	$47.60_{\pm 1.4}$
LLaMA 7B (ShareA$_{qkv}$)	135.5M	$41.01_{\pm 0.8}$	LLaMA 13B (ShareA$_{qkv}$)	212.0M	$\mathbf{48.76}_{\pm \mathbf{0.7}}$
LLaMA 7B (ShareA)	89.3M	$40.93_{\pm 0.5}$	LLaMA 13B (ShareA)	139.1M	$48.15_{\pm 0.5}$
LLaMA2 7B *	6898.3M	45.7	LLaMA2 13B *	13266M	53.8
LLaMA2 7B (LoRA)	159.9M	$47.47_{\pm 1.1}$	LLaMA2 13B (LoRA)	250.3M	$55.31_{\pm 0.2}$
LLaMA2 7B (ShareA$_{qkv}$)	135.5M	$47.88_{\pm 0.1}$	LLaMA2 13B (ShareA$_{qkv}$)	212.0M	$\mathbf{55.66}_{\pm \mathbf{0.1}}$
LLaMA2 7B (ShareA)	89.3M	$\mathbf{48.19}_{\pm \mathbf{0.4}}$	LLaMA2 13B (ShareA)	139.1M	$55.53_{\pm 0.3}$

Table 2: LLaMA and LLaMA2, ranging from 7B to 13B, are fine-tuned using different sharing approaches on the Alpaca datasets and evaluated on the MMLU 5 shot benchmark. The configuration runs is based on the setup described in (Dettmers et al., 2023).* indicates numbers published in prior works, reported by (Xu et al., 2023).

Method	# Params	BLUE	NIST	MET	ROUGE-L	CIDEr
GPT-2 M (FT)*	354.92M	68.2	8.62	46.2	71.0	2.47
GPT-2 M (AdapterL)*	0.37M	66.3	8.41	45.0	69.8	2.40
GPT-2 M (AdapterL)*	11.09M	68.9	8.71	46.1	71.3	2.47
GPT-2 M (PreLayer)*	0.35M	**69.7**	**8.81**	46.1	71.4	2.49
GPT-2 M (LoRA)	0.35M	$69.5_{\pm .7}$	$8.74_{\pm .08}$	$46.56_{\pm .2}$	$71.51_{\pm .3}$	$2.50_{\pm .01}$
GPT-2 M (ShareB)	0.20M	$67.1_{\pm .7}$	$8.55_{\pm .09}$	$45.12_{\pm .4}$	$69.45_{\pm .6}$	$2.37_{\pm .01}$
GPT-2 M (ShareA)	0.20M	$\mathbf{69.7}_{\pm \mathbf{.4}}$	$8.75_{\pm .05}$	$\mathbf{46.60}_{\pm \mathbf{.1}}$	$\mathbf{71.63}_{\pm \mathbf{.1}}$	$\mathbf{2.51}_{\pm \mathbf{.01}}$
GPT-2 L (FT)*	774.03M	68.5	8.78	46.0	69.9	2.45
GPT-2 L (AdapterL)*	0.88M	69.1	8.68	46.3	71.4	2.49
GPT-2 L (AdapterL)*	23.00M	68.9	8.70	46.1	71.3	2.45
GPT-2 L (PreLayer)*	0.77M	**70.3**	**8.85**	46.3	71.7	2.47
GPT-2 L (LoRA)	0.77M	$69.8_{\pm .4}$	$8.80_{\pm .04}$	$\mathbf{46.69}_{\pm \mathbf{.1}}$	$71.71_{\pm .3}$	$\mathbf{2.52}_{\pm \mathbf{.01}}$
GPT-2 L (ShareB)	0.39M	$69.7_{\pm .2}$	$8.80_{\pm .01}$	$46.17_{\pm .3}$	$70.94_{\pm .5}$	$2.49_{\pm .02}$
GPT-2 L (ShareA)	0.39M	$70.0_{\pm .1}$	$8.83_{\pm .03}$	$46.60_{\pm .1}$	$\mathbf{71.74}_{\pm \mathbf{.1}}$	$\mathbf{2.52}_{\pm \mathbf{.02}}$

Table 3: GPT-2 medium (M) and large (L) with different adaptation methods on the E2E NLG Challenge. For all metrics, higher is better. LoRA ShareA outperforms several baselines with comparable or fewer trainable parameters. * indicates numbers published in prior works.

age score of 87.0 on GLUE compared to IA^3's 86.0. Furthermore, under a similar trainable parameter size, ShareA demonstrates better performance than LoRA-FA. For example, ShareA achieves an average GLUE score of 89.3 with 0.4M parameters on RoBERTa-large, surpassing LoRA-FA's score of 88.5 with the same parameter count.

Model Adaptability ShareLoRA demonstrates superior adaptability across a diverse range of tasks and model sizes. In experiments with RoBERTa-base model on the GLUE benchmark shown in Table 1, ShareA exhibits particular strength on smaller datasets that are typically prone to overfitting. Specifically, on tasks such as MRPC, CoLA, and RTE, ShareA achieves performance gains of 0.2% to 0.5%. These improvements are especially noteworthy given that these datasets have generally reached full convergence under standard training configurations (Hu et al., 2021), suggesting ShareLoRA's ability to extract additional performance even in challenging scenarios.

ShareA further showcases enhanced transfer learning capabilities. When fine-tuning on adaptive tasks like MRPC, RTE, and STS-B using the best MNLI checkpoint, ShareA consistently performs on par with or outperforms LoRA. Notably, ShareA outperforms other PEFT methods in this transfer learning scenario as well. For instance, on the RTE task, ShareA, with 0.16M parameters for RoBERTa-base, achieves a score of 87.1, significantly surpassing Prefix-tuning's 54.51 as shown in Table 1. ShareA also demonstrates superior performance when compared to methods with similar trainable parameter sizes, such as BitFit with 0.1M parameters and LoRA-FA with 0.15M parameters. This highlights ShareA's efficiency in parameter utilization and its ability to extract better performance from a given parameter budget, particularly in transfer learning scenarios.

The adaptability of ShareLoRA extends beyond smaller models to larger models as well. As evidenced in Table 2, compared with LoRA, LLaMA2 models fine-tuned with ShareA demonstrate improvements in MMLU scores: the 7B variant improves from 47.47 to 48.19, while the 13B variant shows an increase from 55.31 to 55.53. These results collectively underscore ShareLoRA's efficacy in enhancing model adaptability across vari-

Method	MMLU	ARC (c)	Hellaswarg	GSM8K
LLaMA 7B (LoRA)	41.28	48.49	76.74	2.43
LLaMA 7B (ShareA)	40.67	48.82	76.67	3.16
LLaMA 13B (LoRA)	45.02	**51.34**	79.46	5.79
LLaMA 13B (ShareA)	**46.04**	51.19	**79.53**	**6.17**
LLaMA2 7B (LoRA)	45.68	49.60	77.14	3.21
LLaMA2 7B (ShareA)	47.09	50.14	76.77	6.06
LLaMA2 13B (LoRA)	53.21	51.28	76.59	12.33
LLaMA2 13B (ShareA)	**53.70**	**52.48**	**79.43**	**14.99**

Table 4: Performance of LLaMA models trained on the Alpaca General dataset and tested in a zero-shot of MMLU, ARC Challenge, and Hellaswarg, and in a five-shot of GSM8K, using the lm-eval-harness leaderboard (Gao et al., 2023). This table demonstrates the model's cross-domain adaptability in common sense, reasoning, and mathematics after finetuning on the general dataset.

Method	MATH	MMLU	MMLU Pro
LLaMA3 8B (LoRA)	12.46	65.93	31.80
LLaMA3 8B (ShareA)	**13.24**	**66.35**	**32.55**
LLaMA3.1 8B (LoRA)	**15.36**	65.59	32.86
LLaMA3.1 8B (ShareA)	15.10	**65.78**	**33.69**

Table 5: Performance of LLaMA3 trained on the MATH dataset (Hendrycks et al., 2021b) and evaluated in a zero-shot of MATH and in five-shot of MMLU and MMLU-Pro. It highlights the model's ability to maintain cross-domain adaptability in common sense and reasoning domains after finetuning on mathematics.

Method	# Params	MMLU	Method	# Params	MMLU
LLaMA 7B (QLoRA)*	79.9M	38.8	LLaMA 13B (QLoRA)*	125.2M	47.8
LLaMA 7B (QLoRA)*	79.9M	39.96	LLaMA 13B (QLoRA)*	125.2M	47.29
LLaMA 7B (QLoRA)	79.9M	40.63 ± 0.9	LLaMA 13B (QLoRA)	125.2M	47.13 ± 0.9
LLaMA 7B (QShareA$_{qkv}$)	67.7M	40.63 ± 0.5	LLaMA 13B (QShareA$_{qkv}$)	106.0M	$\mathbf{47.36 \pm 0.7}$
LLaMA 7B (QShareA)	44.6M	$\mathbf{41.11 \pm 0.2}$	LLaMA 13B (QShareA)	69.5M	47.17 ± 0.8

Table 6: Performance comparison of LLaMA 7B and 13B with QLoRA and QShareA under the same configuration of (Dettmers et al., 2023), $*$ is similar experiment results collected from prior work (Xu et al., 2023)

ous scales and task domains.

Cross-domain Robustness ShareLoRA demonstrates robustness across diverse domains and tasks, particularly in zero-shot and few-shot learning scenarios. As illustrated in Table 4, ShareA consistently outperforms LoRA across multiple benchmarks. For the LLaMA2 13B model, ShareA achieves notable improvements: on MMLU, it scores 53.70 while LoRA achieves 53.21; on ARC Challenge, ShareA reaches 52.48 compared to LoRA's 51.28; on Hellaswag, ShareA attains 79.43 while LoRA scores 76.59; and on GSM8K in the 5-shot setting, ShareA achieves 14.99 versus LoRA's 12.33.

To assess the robustness and knowledge retention during domain-specific fine-tuning, we deploy the LLaMA3 and LLaMA3.1 models on the MATH (Hendrycks et al., 2021b) dataset. We evaluate their performance in mathematics and across other domains, such as MMLU and MMLU-Pro, to compare how well these models preserve knowledge, as shown in Table 5. Our findings indicate that both ShareLoRA and LoRA deliver matched performances for directly fine-tuned domains. However, when adapting these fine-tuned models to other evaluation benchmarks, ShareLoRA demonstrates greater robustness, outperforming LoRA. Specifically, in the MMLU-Pro tests, ShareLoRA achieves scores of 33.69 on LLaMA3.1 and 32.55 on LLaMA3, compared to LoRA's scores of 32.86 and 31.80.

The robustness of ShareLoRA extends to quantized models. Table 6 shows that QShareA maintains strong performance even with substantial parameter reduction. In the case of LLaMA 7B, QShareA achieves an MMLU score of 41.11, surpassing QLoRA's score of 40.63. This trend continues with larger models: for LLaMA 13B, QShareA slightly outperforms QLoRA with scores of 47.17 and 47.13 respectively, while using significantly fewer parameters. These performance gains are consistently observed across different model sizes, including LLaMA2 7B and LLaMA 13B, highlighting ShareLoRA's broad applicability and scalability.

6 Analysis and Discussion

Relative Importance of LoRA Components Our experimental findings demonstrate that both LoRA and ShareA consistently outperform ShareB in a variety of classification and generative tasks, across most metrics. Within the LoRA framework, the up-projection matrix B plays a pivotal role by significantly enhancing the dimensionality of the low-rank representation. Consequently, it is both

practical and justifiable to share the less critical module, LoRA A, while retaining the integrity of B. However, sharing both matrices A and B simultaneously tends to compromise too much critical information. Particularly in generative tasks, opting to share component A instead of B within the ShareLoRA framework is strategically beneficial, as seen in Table 3. This is because expanding the intermediate dimension proves more crucial and challenging than compressing high-dimensional features in complex generative scenarios.

Sharing Attention QKV vs. Sharing All The distinction between sharing the self-attention mechanism and all linear modules exists on MLP components like gates and up/down projections. This leads to a discrepancy in trainable parameters between LoRA's A and B. The strategic choice involves deciding whether to uniformly share weights across all layers (ShareA) or selectively share them, such as only for the down projection (ShareAB) while maintaining unique weights for other components like the up projection and gates. Preliminary results in Appendix Figure 4 suggest that selective sharing, particularly of the QKV matrices in Share$_{qkv}$, provides an effective balance by aligning closely with both ShareA and LoRA , potentially mitigating overfitting risks.

Memory Footprint In the context of smaller models like RoBERTa and GPT-2, ShareA yields minimal parameter savings, which is negligible given modern GPU capacities. However, with larger models like LLaMA, ShareA demonstrates more substantial reductions. Specifically, the LLaMA 7B and 13B models cut down approximately 60 million and 110 million trainable parameters, respectively, when compared to the LoRA architecture. This leads to substantial efficiency gains, reducing both computational footprint and disk storage needs.

As depicted in Figure 2 in the Appendix, ShareA achieves a memory footprint reduction of 1.8GB and approximately a 2% increase in training speed, while ShareAB can save around 4GB with 4% training speed up. The confidence intervals in Table 2 illustrate that ShareA not only improves performance but also increases robustness over standard LoRA, underscoring the practical advantages of ShareLoRA in LLMs.

SVD Analysis of LoRA and ShareA Weights We conducted a Singular Value Decomposition (SVD) analysis on the weights of LLaMA 13B for both LoRA and ShareA, as shown in Figure 6 in the Appendix. The results reveal distinct patterns in their singular value distributions across layers. LoRA weights exhibit a sharp decrease in singular values, indicating a concentration of information in a few dominant components. This could lead to specialization but might also increase the risk of overfitting. In contrast, ShareA weights show a smoother, more gradual decrease in singular values, suggesting a more balanced distribution of information among components. This balanced distribution contributes to ShareA's enhanced adaptability and generalization capability across different tasks.

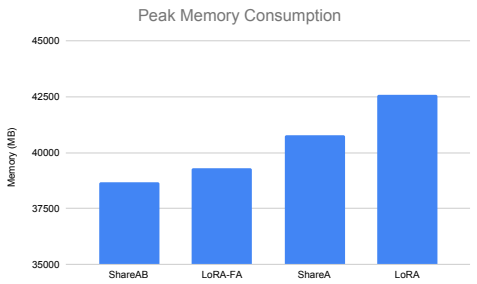

Figure 2: Memory Consumption required for LLaMA 13B

These findings provide insight into why ShareA may offer improved robustness and cross-task performance compared to standard LoRA. The more uniform distribution of singular values in ShareA suggests that it captures a broader range of features, leading to better generalization across various tasks and domains.

7 Conclusion

In this paper, we introduce ShareLoRA, an optimization of the LoRA architecture that shares either the up or down projection across different layers. ShareLoRA significantly reduces the number of trainable parameters by at least half relative to the original LoRA and shows improved performance on fully converged datasets. Through extensive experimentation with NLU, NLG, and zero-shot tasks on models varying from millions to billions of parameters, ShareLoRA provides an optimal balance between computational efficiency and robust performance. By sharing all linear components or focusing solely on self-attention mechanisms, ShareLoRA potentially reduces overfitting risks while maintaining high adaptability and robustness across various domains.

8 Limitation

The limitations of ShareLoRA are primarily in its convergence speed and practical applications. ShareAB and ShareB tend to converge more slowly compared to LoRA, though ShareA shows a convergence rate that is largely competitive with LoRA on smaller datasets, with only a slight lag on larger datasets. This indicates that ShareA is quite adept at easily converged datasets and effectively mitigating near-overfitting scenarios.

Regarding the practical application of GPUs, ShareLoRA introduces some complexities in the parallel training process on multiple GPUs. This is primarily due to the need for consistent synchronization of the Shared Module, once it is replicated across various GPUs at every computational step.

References

Tom B. Brown and Benjamin Mann et.al. 2020. Language models are few-shot learners.

Arnav Chavan, Zhuang Liu, Deepak Gupta, Eric Xing, and Zhiqiang Shen. 2023. One-for-all: Generalized lora for parameter-efficient fine-tuning.

François Chollet. 2019. On the measure of intelligence.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Raj Dabre, Atsushi Fujita, and Chenhui Chu. 2019. Exploiting multilingualism through multistage fine-tuning for low-resource neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1410–1416, Hong Kong, China. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Chowdhery et.al. 2022. Palm: Scaling language modeling with pathways.

Touvron et.al. 2023a. Llama 2: Open foundation and fine-tuned chat models.

Touvron et.al. 2023b. Llama: Open and efficient foundation language models.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning.

Shwai He, Run-Ze Fan, Liang Ding, Li Shen, Tianyi Zhou, and Dacheng Tao. 2023. Mera: Merging pre-trained adapters for few-shot learning.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Jordan Hoffmann and Sebastian Borgeaud et.al. 2022. Training compute-optimal large language models.

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

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

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. Lorahub: Efficient cross-task generalization via dynamic lora composition.

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.

Tao Lei, Junwen Bai, Siddhartha Brahma, Joshua Ainslie, Kenton Lee, Yanqi Zhou, Nan Du, Vincent Y Zhao, Yuexin Wu, Bo Li, Yu Zhang, and Ming-Wei Chang. 2023. Conditional adapters: Parameter-efficient transfer learning with fast inference. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

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

Xiang Lisa Li and Percy Liang. 2021b. Prefix-tuning: Optimizing continuous prompts for generation.

Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. 2023. Relora: High-rank training through low-rank updates.

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2020. Exploring versatile generative language model via parameter-efficient transfer learning.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

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.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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. *Journal of Machine Learning Research*, 21(140):1–67.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. Adapterdrop: On the efficiency of adapters in transformers.

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.

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.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark.

BigScience Workshop, :, Teven Le Scao, and Angela Fan et.al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Yuqing Xie, Wei Yang, Luchen Tan, Kun Xiong, Nicholas Jing Yuan, Baoxing Huai, Ming Li, and Jimmy Lin. 2020. Distant supervision for multi-stage fine-tuning in retrieval-based question answering. In *Proceedings of The Web Conference 2020*, WWW '20, page 2934–2940, New York, NY, USA. Association for Computing Machinery.

Lingling Xu and Weiming Wang. 2023. Improving aspect-based sentiment analysis with contrastive learning. *Natural Language Processing Journal*, 3:100009.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models.

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*.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning.

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

A Hyperparameters

In our study, we limits the extent of hyperparameter optimization in order to maintain consistency with prior research (Hu et al., 2021; Dettmers et al., 2023; Mahabadi et al., 2021; Gao et al., 2023), facilitating a direct comparison. Furthermore, we aims to investigate the behaviors of underfitting and overfitting across different scenarios using the LoRA and ShareLoRA approaches applied to various model size.

Specifically, under the current training setup, both LoRA and ShareLoRA exhibit signs of non-convergence when applied to the LLaMA 7B model. On the other hand, LoRA demonstrates clear overfitting when used with the LLaMA2 13B model, suggesting that the model training has gone beyond the point of optimal generalization.

For the models LLaMA 13B and LLaMA 2 7B, their performances are comparable. Both models reach a point of convergence and display fluctuations around this state, indicating that they are fully trained. It helps us understand the differing impacts of LoRA and ShareLoRA on these models under a set of reasonable training configurations.

The hyperparameter setting for RoBERTa is in Table 7 and for LLaMA are in Table 9 and 10. The number of trainable parameters in Table 6, should remain consistent between QLoRA and LoRA for LLaMA 7B and 13B in Table 2, as both models utilize BFloat16. However, the reduced number of trainable parameters is influenced by the implementation described in (Dettmers et al., 2023), which reduces the trainable parameters by half when quantizing to 4 bits. This is also reported the same by (Xu et al., 2023), and we maintain this parameter count to ensure consistency.
We conducted five experiments with Roberta and GPT-2, and three experiments for all tasks related to LLaMA using different seeds. The results presented are all averages.

B LLaMA Performance Analysis

In Figures 3 and 4, we present the Dev Set performance changes for both LLaMA and LLaMA2 models, ranging from 7B to 13B, to observe the differences in performance over steps. The results demonstrate that ShareA and ShareA$_{qkv}$ configurations offer several advantages over their counterparts, as discussed in Section 6.

For both the 7B and 13B models, ShareA and ShareA$_{qkv}$ configurations maintain higher average accuracy compared to the traditional LoRA setup. Specifically, ShareA demonstrates consistent performance improvements, particularly in the stability of accuracy over different steps. This indicates that ShareA is more robust and less prone to fluctuations compared to LoRA.

The analysis in Figure 3 further enriches our results by incorporating confidence intervals which map the performance stability of LoRA, QLoRA, ShareA, and QShareA. From these plots, it is evident that while LoRA occasionally outperforms QLoRA, the overall performance trends of LoRA and QLoRA are closely aligned in LLaMA 7B. In particular, for the LLaMA 13B, the performance of ShareA and QShareA after 5000 steps is completely superior than LoRA and QLoRA. It is crucial to highlight that both LoRA and QLoRA display larger fluctuations in performance compared to ShareA and QShareA, underscoring a potentially greater variability in model outcomes across different experimental seeds.

C Convergence Analysis

In Figure 5, we analyze the convergence trends across both the MNLI and CoLA datasets for the RoBERTa-large model, demonstrating differing behaviors among the sharing strategies and others. Notably, while ShareA begins with slightly lower performance compared to LoRA, it progressively matches LoRA's accuracy on the MNLI dataset. ShareB and ShareAB, in contrast, consistently underperform relative to both LoRA and ShareA. This pattern is similarly observed with the CoLA dataset, where ShareA's performance is robust, closely competing with LoRA. Both ShareB and ShareAB are worse than LoRA alone.

At the same time, LoRA-FA only reaches performance levels comparable to ShareB, lagging behind both ShareA and LoRA. This suggests that ShareA not only sustains competitive convergence capabilities but also outperforms LoRA-FA in terms of robustness and eventual alignment with LoRA's top performance.

In term of training loss, all models exhibit a similar declining trend over the training epochs. However, ShareA distinguishes itself by slightly lagging behind LoRA initially in terms of speed of convergence but substantial surpassing both ShareB and LoRA-FA overall. This differential suggests that ShareA offers a balanced approach, effectively managing a slower initial convergence for consis-

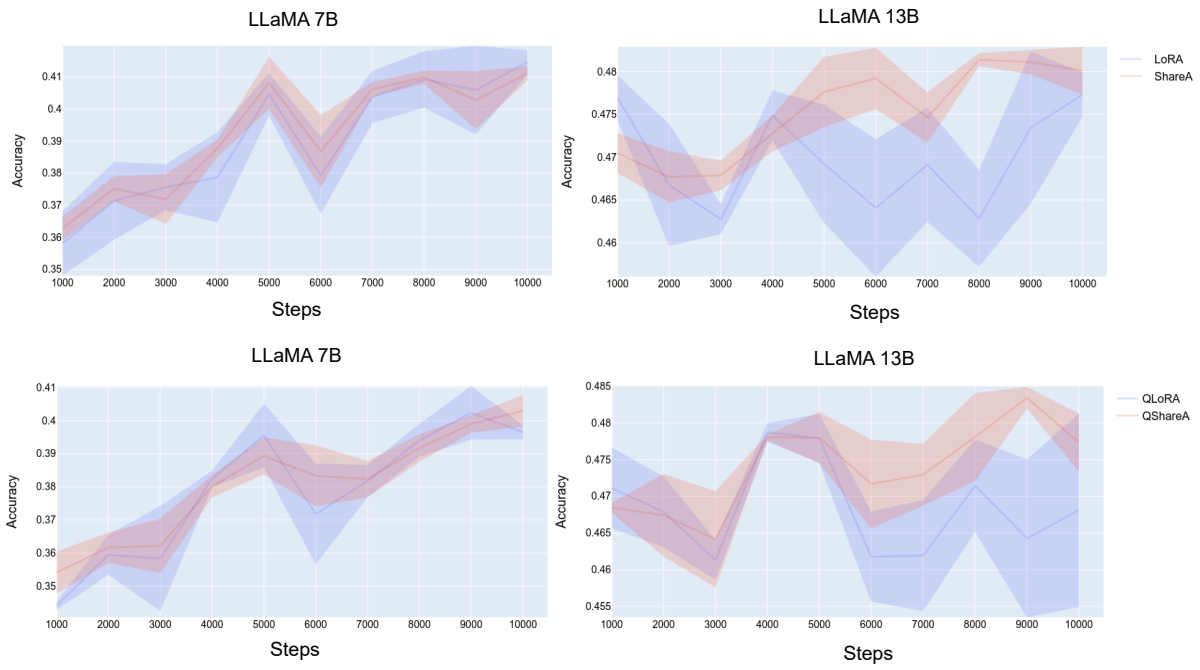

Figure 3: LLaMA 7B & 13B on LoRA / ShareA (upper) and on QLoRA / QShareA (down) MMLU Dev Performance with the standard deviation error distribution of different seeds

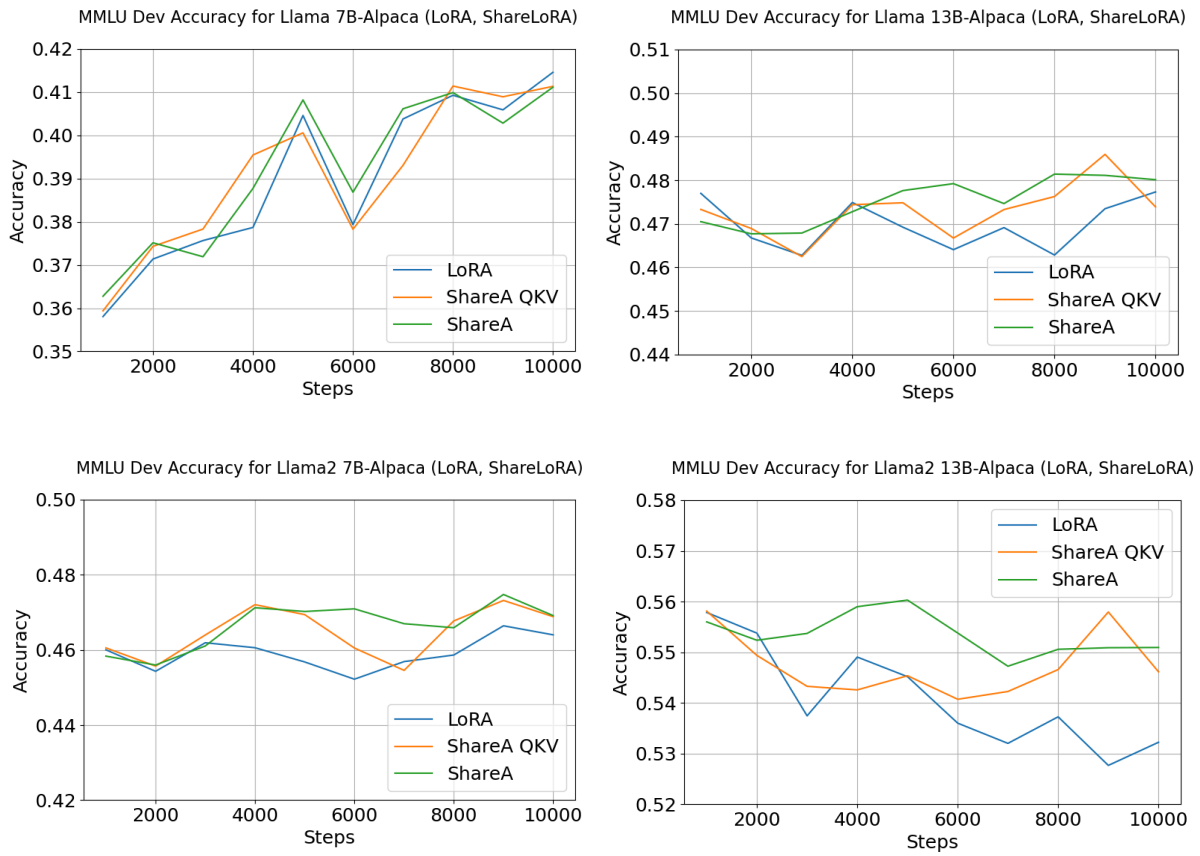

Figure 4: Average Performance Plot for Various LLaMA Models on the Alpaca-MMLU Dev Dataset

tent long-term gains.

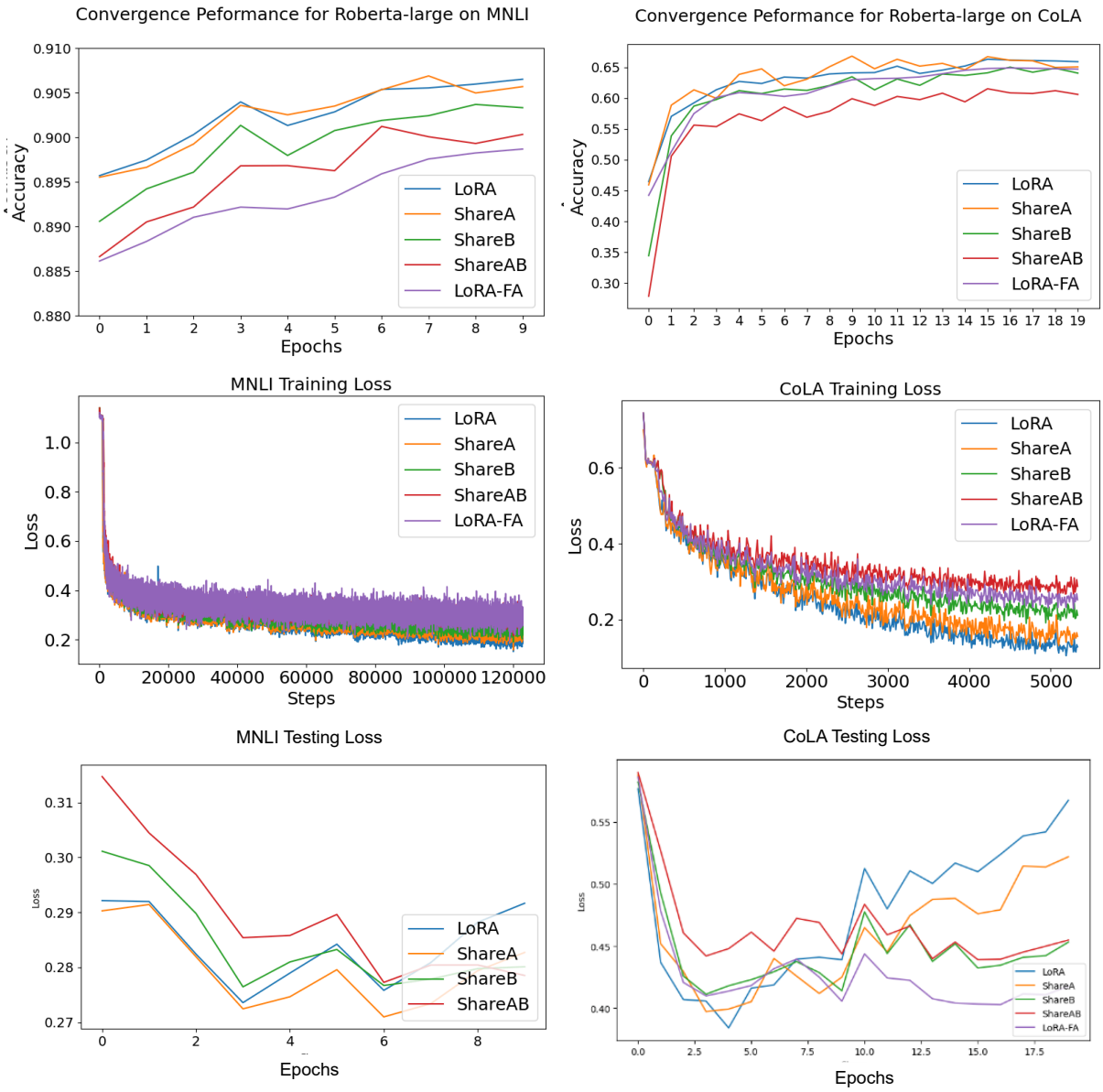

Figure 5: Convergence Performance for MNLI and CoLA datasets

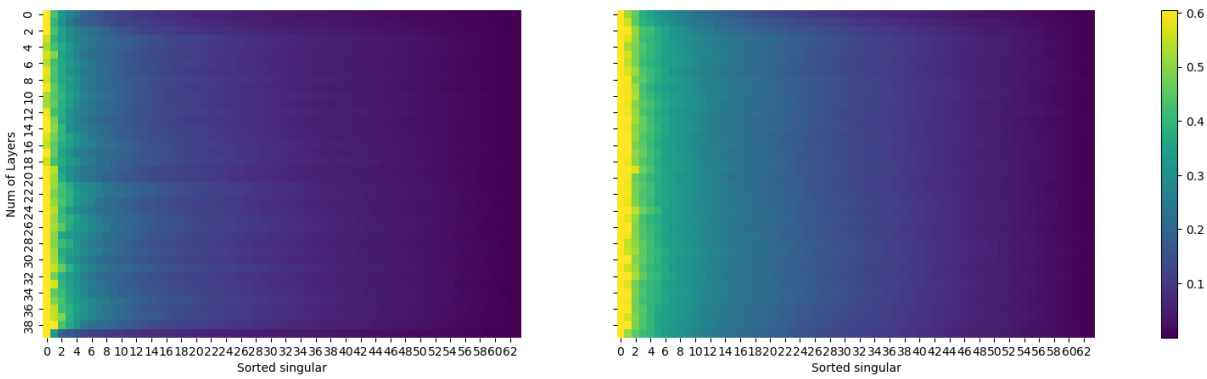

Figure 6: Distribution of Singular Values for LLaMA 13B: SVD Decomposition Analysis of LoRA (left) and ShareA (right) across All Layers.

Method	Dataset	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
	Optimizer				AdamW				
	Warmup Ratio				0.06				
	LR Schedule				Linear				
RoBERTa base ShareLoRA	Batch Size (per device)	16	16	16	32	32	16	32	16
	# Epochs	30	60	30	80	25	25	80	40
	Learning Rate	5E-04	5E-04	4E-04	4E-04	4E-04	5E-04	5E-04	4E-04
	LoRA Config.				$r_q = r_v = 8$				
	LoRA α				8				
	Max Seq. Len.				512				
	seed				0,1,2,3,4				
RoBERTa large ShareLoRA †	Batch Size (per device)				4				
	# Epochs	10	10	20	20	10	20	20	10
	Learning Rate	3E-04	4E-04	3E-04	2E-04	2E-04	3E-04	4E-04	2E-04
	LoRA Config.				$r_q = r_v = 8$				
	LoRA α				8				
	Max Seq. Len.				512				
	seed				0,1,2,3,4				

Table 7: Configuration and training details for RoBERTa base LoRA on different datasets.

Dataset	E2E Challege
Optimizer	AdamW
Weight Decay	0.01
Dropout Prob	0.1
Batch Size (per device)	8
# Epochs	5
Warmup Steps	500
Learning Rate Schedule	Linear
Label Smooth	0.1
Learning Rate	0.002
Adaptation	$r_q = r_v = 4$
LoRA α	32
Beam Size.	10
Length Penalty	0.9
no repeat ngram size	4

Table 8: Configuration and training details for GPT-2 LoRA on E2E Challenge

Model	Parameters	Batch size	LR	Steps	Source Length	Target Length	LoRA r	LoRA α
LLaMA1 & 2	7B	16	2e-4	10000	384	128	64	16
LLaMA1 & 2	13B	16	2e-4	10000	384	128	64	16
LLaMA3 & 3.1	8B	16	2e-5	5000	384	128	64	16

Table 9: Training hyperparameters for LLaMA and QLLaMA.

Parameters	MMLU Source Length	Temperature	Top P	Beam size
7B	2048	0.7	0.9	1
13B	2048	0.7	0.9	1

Table 10: Evaluation hyperparameters for LLaMA and QLLaMA.