ASLoRA: Adaptive Sharing Low-Rank Adaptation Across Layers

Junyan Hu[1], Xue Xiao[2], Mengqi Zhang[1], Xiao Chen[2], Zhaochun Ren[3], Zhumin Chen[1], Pengjie Ren[1]*

[1]Shandong University, Qingdao, China
[2]Inspur Cloud Information Technology Co.,Ltd
[3]Leiden University, Leiden, The Netherlands
hujunyan@mail.sdu.edu.cn
{renpengjie,mengqi.zhang,chenzhumin}@sdu.edu.cn
xiaoxue@inspur.com, chenyao@inspur.com

Abstract

As large language models (LLMs) grow in size, traditional full fine-tuning becomes increasingly impractical due to its high computational and storage costs. Although popular parameter-efficient fine-tuning methods, such as LoRA, have significantly reduced the number of tunable parameters, there is still room for further optimization. In this work, we propose ASLoRA, a cross-layer parameter-sharing strategy combining global sharing with partial adaptive sharing. Specifically, we share the low-rank matrix A across all layers and adaptively merge matrix B during training. This sharing mechanism not only mitigates overfitting effectively but also captures inter-layer dependencies, significantly enhancing the model's representational capability. We conduct extensive experiments on various NLP tasks, showing that ASLoRA outperforms LoRA while using less than 25% of the parameters, highlighting its flexibility and superior parameter efficiency. Furthermore, in-depth analyses of the adaptive sharing strategy confirm its significant advantages in enhancing both model flexibility and task adaptability.

1 Introduction

The advent of large language models (LLMs) like GPT-3.5 Turbo (OpenAI, 2023), Gemini (Anil et al., 2023), and LLaMA3 (Dubey et al., 2024) marks a breakthrough in NLP. However, due to their massive parameters, fully fine-tuning these models for specific tasks is expensive, especially as model sizes grow (Brown et al., 2020). In response, parameter-efficient fine-tuning (PEFT), such as adapter (Houlsby et al., 2019; Hu et al., 2022) and Prefix Tuning (Li and Liang, 2021), have gained popularity. These methods fine-tune only a small subset of parameters, reducing storage and computation demands significantly.

*Corresponding author

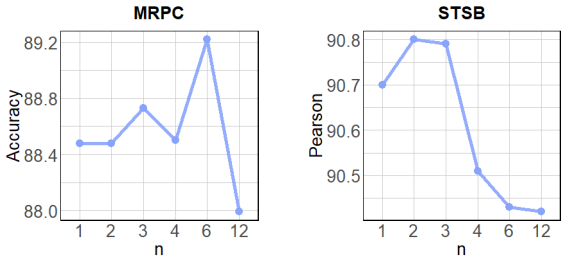

Figure 1: The pre-experiment on the MRPC and STSB datasets. We let A be shared in all layers and make adjacent n layers share the same B, where $n = 3$ means that every 3 adjacent layers share the same B.

As a popular method of parameter-efficient fine-tuning (PEFT), LoRA (Hu et al., 2022) introduces two low-rank matrices, A and B, whose product represents the update to the weight matrix, i.e., $W_0 + \Delta W = W_0 + BA$. Given that the ranks of A and B are significantly smaller than the original model dimensions, this approach greatly reduces the number of tunable parameters. Moreover, LoRA directly adds the product of the low-rank matrices to the weight matrix, without introducing additional inference latency. Despite its excellent performance, LoRA still requires a substantial number of parameters.

To address this issue, several studies have explored combining parameter sharing with LoRA. For instance, VeRA (Kopiczko et al., 2024) shares randomly initialized matrices A and B across all layers and freezes their parameters while introducing trainable scaling vectors between them to reduce the number of tunable parameters. However, their weight-freezing strategy limits model expressiveness. Subsequently, Tied LoRA (Renduchintala et al., 2024) alleviates these issues by allowing trainable matrices to be shared across layers, but its binding mechanism restricts applicability to weights of varying shapes. ShareLoRA (Song et al.,

2024) introduces an asymmetric sharing mechanism where the matrix A is shared across all layers, while the matrix B is not. Although this approach significantly reduces the number of parameters by reusing A across layers, it is relatively simplistic and lacks a detailed analysis of whether B could also benefit from sharing.

To this end, we investigate the effects of partially sharing matrix B while maintaining full sharing of matrix A across all layers. We conduct preliminary experiments and show the results in Figure 1. We observe that different sharing strategies yield different results, and in some cases, a smaller parameter size can lead to better performance. This suggests potential redundancies in B, pointing to a fine-grained sharing approach that reduces parameters while enhancing performance. Inspired by this, we propose a fine-tuning approach called **A**daptive **S**haring **Lo**w-**R**ank **A**daptation Across Layers (ASLoRA). We divide the training process into three stages: shared training, adaptive merging, and final optimization. In the shared training stage, the matrix A is shared across all layers to capture global information while reducing the number of trainable parameters by half. Meanwhile, matrix B remains unshared to capture the unique information of each layer. In the adaptive merging stage, to eliminate redundancy among the B matrices of different layers and further reduce parameters, we merge these matrices based on their similarity. In the final optimization stage, the merged model structure is retained and further trained to ensure convergence and optimal performance. Compared to LoRA, ASLoRA combines global and local sharing, using fewer parameters and effectively alleviating the overfitting problem. We conduct comprehensive experiments on multiple tasks and models, using RoBERTa-base for natural language understanding (NLU) tasks and LLaMA-2-7B for instruction tuning tasks. The experimental results show that ASLoRA achieves better performance with fewer parameters than LoRA, outperforming the baseline models across all instruction-following datasets. In summary, our contributions are as follows:

- We experiment with different ways of sharing matrix B while maintaining full sharing of matrix A across all layers. We find that some strategies with fewer parameters perform better.
- We propose a parameter-sharing approach, ASLoRA, which combines global sharing with partial adaptive sharing to further enhance parameter efficiency.
- We compare ASLoRA with existing methods across multiple tasks, showing that it achieves higher parameter efficiency and superior performance.

2 Related Work

2.1 Parameter-Efficient Fine-Tuning

As transformer models scale up and downstream tasks increase, full fine-tuning poses significant computational challenges. To address this, parameter-efficient fine-tuning methods have emerged, which update only a small portion of the model's parameters to achieve performance comparable to full fine-tuning. Prompt Tuning (Shin et al., 2020; Chen et al., 2022) introduces task-specific prompts to adjust the model precisely, Adapter Tuning (Houlsby et al., 2019) adds lightweight adapters between model layers to drastically reduce resource consumption, and Prefix-Tuning (Li and Liang, 2021) prepends a continuous, task-specific vector sequence to the model's input. While these methods have shown remarkable effectiveness, fine-tuning large models still demands substantial computational resources, especially in resource-constrained environments.

2.2 Low-Rank Adaptation

Low-Rank Adaptation (LoRA) (Hu et al., 2022) using the product of two low-rank matrices to approximate weight updates. It is widely adopted due to its simplicity and lack of inference delay. Current improvements to LoRA focus primarily on enhancing performance and reducing parameter count. AdaLoRA (Zhang et al., 2023b) and IncreLoRA (Zhang et al., 2023a) improve LoRA by introducing higher ranks for more critical modules, but varying ranks across layers complicate multi-LoRA deployment. VeRA (Kopiczko et al., 2024) reduces parameter count by sharing a frozen random matrix across layers and training two low-parameter vectors, but it affected performance. MELoRA (Ren et al., 2024) connects multiple mini LoRAs to reduce parameter count while maintaining rank, at the cost of increased time complexity. PRoLoRA (Wang et al., 2024) introduces shared and rotated enhancements within LoRA, effectively reducing parameters but remaining limited to internal LoRA interactions, thus unable to capture inter-layer dependencies. In contrast, our method employs a cross-layer parameter-

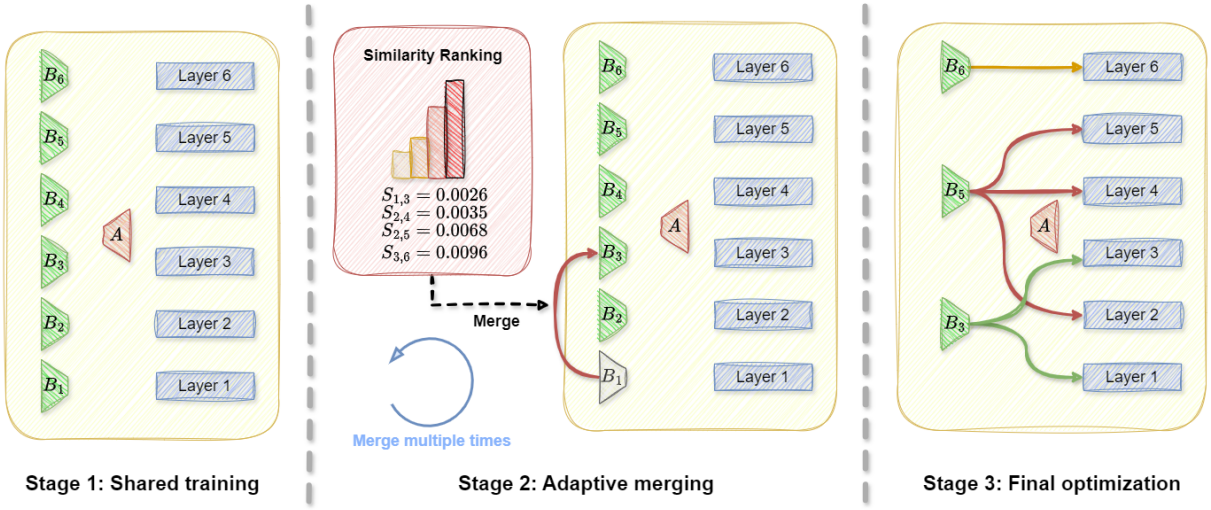

Figure 2: Illustration of ASLoRA, we present a six-layer model. First, all layers share matrix A and enter the shared training phase on the left. The center shows the adaptive merging process, where the most similar B matrices are merged each time based on their pairwise similarity. After several merges, the model moves to the final optimization phase on the right, with partial sharing of B completed.

sharing mechanism, effectively mitigating these limitations.

2.3 Parameter Sharing

Parameter sharing is widely used to reduce model memory requirements. Universal Transformer (Dehghani et al., 2019) reduces parameter count by sharing all layers. Takase and Kiyono introduced three cross-layer parameter sharing strategies that lower both parameters and computation demands. Subformer (Reid et al., 2021) achieves significant parameter reduction without performance loss through middle-layer sharing and embedding factorization. LightFormer (Lv et al., 2023) uses SVD-based weight transfer and low-rank factorization for model compression and acceleration, while Relaxed Recursive Transformers (Bae et al., 2024) improves inference speed through cross-layer sharing via a recursive structure. Differently, our method focuses on PEFT scenarios, aiming to improve the parameter efficiency of LoRA models rather than directly optimizing transformer models.

3 Method

In this section, we will introduce ASLoRA, a adaptive method for sharing parameters across layers. In simple terms, we let A share across all layers, and let B share adaptively during training, reducing parameters while learning the information associated with each layer. We show our structure in Figure 2.

3.1 Preliminaries on Low-Rank Adapter

LoRA freezes the original weight matrix W_0 and decomposes the weight update ΔW into two low rank matrices B and A. The forward propagation process is shown in equation 1:

$$h = W_0 x + \Delta W x = W_0 x + BAx. \quad (1)$$

Here, $W_0 \in \mathbb{R}^{d \times d}$ is the pretrained weight matrix, h is the output vector, $x \in \mathbb{R}^d$ is the input vector and $\Delta W = BA$ is the increment matrix during fine-tuning, where $A \in \mathbb{R}^{d \times r}$ and $r \ll d$. During training, A is initialized with a Gaussian distribution, and B is initialized as a zero matrix to ensure the initial increment $BA = 0$.

3.2 Shared Training

To reduce parameters while capturing global information across layers and local details for each layer, we need to share either A or B. Considering that LoRA randomly initializes A, it means that A is different for each layer, while B is initialized to 0, meaning that B is the same across all layers. Therefore, sharing A while not sharing B ensures that each layer's B has the same initialization value, which facilitates measuring the changes in B. So we share A across all layers and use a separate B_i for each layer. The forward propagation process is shown in equation 2:

$$h_i = W_i x + B_i A x, \quad (2)$$

where i is the layer index of the model, h_i is the output of layer i, B_i represents the B of the i-th

layer. This equation indicates that the weight variation ΔW_i for each layer is obtained by matrix A using the corresponding B_i. Shared A is consistent across all layers, reducing redundancy in training and memory requirements. At the same time, the independent B_i of each layer makes specific adjustments to the output to achieve differentiated feature transformation.

3.3 Adaptive Merging

After completing the T_s steps of shared training, the model has learned knowledge of different layers through B. In order to reduce the redundancy of B and further reduce the parameters, we perform an adaptive merge of different B. In simple terms, we calculate the pairwise similarity between the B matrices and merge B with the highest similarity every m steps.

Average Weights. If we directly use the B of step t for similarity calculation, we would only observe the value of B at the current step and fail to measure the overall changes in B during the training phase. Therefore, we introduce the average weight to measure similarity. Specifically, the weight at step t is equal to the average weight of the previous t steps:

$$\overline{B_i^t} = \frac{1}{t}\sum_{k=1}^{t} B_i^k. \tag{3}$$

Here, i is the model layer index, $\overline{B_i^t}$ is the average weight of the B for the layer i, B_i^k is the weight of step k of B, t is the current step. By using average weights, we can better capture the overall B_i information from the previous step, reducing randomness.

Similarity Calculation. The L2 norm can effectively measures the overall distance between vectors and penalizes larger differences more significantly, so we use it to measure the similarity between the two pairs of B at each layer, specifically:

$$S_{i,j}^t = \left\| \overline{B_i^t} - \overline{B_j^t} \right\|_2 = \sqrt{\sum_{k=1}^{n} (b_{i,k}^t - b_{j,k}^t)^2}, \tag{4}$$

where $S_{i,j}^t$ is the similarity between layer i and layer j matrices B, $b_{i,k}^t$ represents each element of layer i matrix B. By using the L2 norm, we can effectively measure pairwise similarities between B-matrices and rank these similarities. From the equation 4, it can be seen that a smaller $S_{i,j}^t$ indicates a higher similarity. Each time, the two B-matrices with the highest similarity are selected for merging.

Weight Merging. Considering that the upper layers of the model contain more complex information (Zhang et al., 2023b), we make the lower layers use the B of the upper layers when merging. This ensures that more useful information is preserved after merging.

3.4 Final Optimization

After completing the merging of B, the model enters the final optimization phase. A remains shared across all layers, while B has undergone partial merged sharing. As a result, some layers share the same B, denoted as $\tilde{B}(i)$, representing the B used in the i-th layer. The forward propagation formula is as follows:

$$h_i = W_i x + \tilde{B}(i) A x. \tag{5}$$

After this stage of training, the model has successfully converged. We summarize the detailed algorithm in Algorithm 1.

Algorithm 1 : ASLoRA. T is the total steps, T_s is the number of steps that start merging, m is the interval between merges, N is the number of merges.

Input: T_s, m, N

```
Share A across all layers as equation (2)
for t = 1, ..., T do
    if N > 0 then
        Update B̄_i by equation (3)
        if t > T_s and (t − T_s)%m == 0 then
            Calculate all S^t by equation (4)
            Sort all S^t and find the minimal S^t_{i,j}
            Merge B_i and B_j, N ← N − 1
        end if
    end if
end for
```

As shown in Algorithm 1, we first share A across all layers and train the model according to equation 2, allowing B to learn the information of each layer during this phase. After completing T_s steps of training, we calculate the pairwise similarity between adjacent layers every m steps, and merge the two layers with the lowest similarity. This process is repeated until N merges are completed. Subsequently, we continue to train based on equation 5.

3.5 Advantage Analysis

Global sharing A has high adaptability. Because LoRA initializes A randomly and B with zeros, this initialization can affect the similarity calculations. By sharing A across all layers, we can effectively eliminate the interference caused by the initialization values. Specifically, all layers' B start with the same value (zero), and each B propagates through the same A. This approach removes the influence of A and the initialization values on B, leading to more reasonable and consistent similarity calculations for B.

Partially sharing B has high flexibility. We share A across all layers to capture the shared knowledge across the entire model. Meanwhile, B is partially shared based on the unique characteristics of each layer. This approach allows ASLoRA to capture both global knowledge and more fine-grained, layer-specific knowledge, providing greater flexibility. Especially when the model has more layers, this adaptive sharing strategy allows for a more flexible distribution of parameters.

ASLoRA has high parameter efficiency. We share A across all layers and merge B during training. This approach can reduce the parameter size by at least half, and as the number of merges increases, the parameter size continues to decrease. As the number of model layers increases, the amount of parameters that can be reduced also increases.

4 Experiments

In this section, we evaluate the performance of ASLoRA in natural language understanding (NLU) and instruction tuning (Chia et al., 2024). For NLU, we use RoBERTa-base (Liu et al., 2019) to test on the GLUE (Wang et al., 2018) dataset. For instruction tuning, we use LLaMA-2-7B as the large language model (LLM) backbone, trained on the alpaca dataset, and evaluate multiple metrics. Finally, we explore the advantages of adaptive merging.

Baselines. We compare ASLoRA with popular parameter-efficient fine-tuning (PEFT) methods. To ensure a fair and comprehensive comparison, we replicate the experimental setups used in previous works and use their reported results. The baseline methods involved are:

- **Full Fine-Tuning (FF)** - The base model is initialized with pre-trained weights and biases, and all parameters undergo gradient updates.
- **Adapter Tuning** - **Adapter$^{\mathbf{H}}$** (Houlsby et al., 2019) inserts two layers of adapters between the self-attention and feed-forward network modules, followed by a residual connection. We also compare three variants: **Adapter$^{\mathbf{L}}$** (Lin et al., 2020), which applies adapter layers only after the MLP module, **Adapter$^{\mathbf{P}}$** (Pfeiffer et al., 2021), which applies adapters after the feed-forward layer, and **Adapter$^{\mathbf{D}}$** (Rücklé et al., 2021), which improves parameter efficiency by removing inactive adapter layers.
- **LoRA** (Hu et al., 2022) - LoRA parameterizes the incremental weight updates using low-rank matrices, making it a state-of-the-art PEFT method.
- **DyLoRA** (Valipour et al., 2023) - This method trains dynamic, search-free LoRA models to select the optimal rank.
- **AdaLoRA** (Zhang et al., 2023b) - Based on singular value decomposition (SVD) and importance scores, AdaLoRA adaptively allocates different ranks to different modules of the model.
- **PiSSA** (Meng et al., 2024) - PiSSA retains LoRA's architecture but initializes the low-rank matrices A and B with the principal components of the original weight matrix W, while storing the remaining components in a residual matrix.

4.1 Natural Language Understanding

Models and Datasets. We validate our approach on the GLUE benchmark (Wang et al., 2018), which includes a variety of natural language understanding (NLU) tasks, such as single-sentence classification, similarity and synonymous sentence tasks, and natural language reasoning tasks. We select RoBERTa-base model (Liu et al., 2019) for evaluation.

Implementation Details. In all experiments, we fine-tune W_Q and W_V, with all data and models downloaded from huggingface. For the GLUE benchmark, we use the LoRA (Hu et al., 2022) configuration, fine-tuning the RoBERTa-base model across 6 datasets. We set the rank to 8, and fine-tune all W_Q and W_V weights as well as the classification heads. For ASLoRA, since the Base model has only 12 layers and supports a maximum of 11 merges, we set the merge count to 7. We provide the hyperparameters in Table 5 in Appendix.

Method	#Params	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
FF	125M	94.8	**90.2**	63.6	92.8	78.7	91.2	85.2
Adpt^{D}	0.3M	94.2	88.5	60.8	93.1	71.5	89.7	83.0
Adpt^{D}	0.9M	94.7	88.4	62.6	93.0	75.9	90.3	84.2
LoRA	0.3M	**95.1**	89.7	63.4	**93.3**	78.4	**91.5**	85.2
AdaLoRA	0.3M	94.5	88.7	62.0	93.1	**81.0**	90.5	85.0
DyLoRA	0.3M	94.3	89.5	61.1	92.2	78.7	91.1	84.5
PiSSA	0.3M	94.7	89.2	**63.8**	92.5	75.5	90.8	84.4
ASLoRA	**0.073M**	94.8	90.0	63.3	93.2	79.8	91.1	**85.4**

Table 1: Performance of various fine-tuning methods with RoBERTa-base models on 6 datasets of the GLUE benchmark. We report the Matthew's correlation coefficient for CoLA, Pearson correlation coefficient for STS-B and accuracy for other tasks. We also report the number of trainable parameters (#Params) for each method. The best results for each dataset are shown in **bold**, the second-best results are underline. Higher is better for all metrics in 6 datasets.

Method	#Params	MMLU	BBH	DROP	HEval	Avg.
w/o FT	-	45.96	32.04	31.55	12.20	30.44
FT	7B	**47.30**	32.72	29.12	12.80	30.49
LoRA	33.6M	45.64	32.40	32.46	15.09	31.40
QLoRA	33.6M	45.40	32.81	28.97	15.24	30.61
AdaLoRA	33.6M	45.96	32.85	31.94	14.02	31.19
ASLoRA	**8.9M**	46.21	**32.96**	**32.47**	**17.68**	**32.33**

Table 2: Results on instruction tuning, we present exact match scores for MMLU, DROP, and BBH, pass@1 for HumanEval(HEval). We also report the average score. With higher values indicating better performance. The best results for each dataset are shown in **bold**, the second-best results are underline.

Results. The results are summarized in Table 1. We report the number of all parameters except the classification header. For ASLoRA, we report the number of parameters after completing the merge. In the case of 7 merges, ASLoRA only uses 24% (0.073M/0.3M) of the parameters, significantly reducing the parameter size, while surpassing all benchmark methods in average score. Although it fails to reach the leading position on a single data set, it ranks second on four data sets (SST-2, MRPC, QNLI and RTE), demonstrating its ability to diversify data sets while reducing the number of parameters. It maintains the advantages of stable performance and excellent generalization ability. Therefore, ASLoRA can significantly reduce the number of parameters and reach or exceed the performance of traditional methods under the condition of limited resources, which fully proves its feasibility and potential.

4.2 Instruction Tuning

Models and Datasets. In this section, we use LLaMA-2-7B as the backbone LLM and train it using the alpaca dataset (Taori et al., 2023), randomly selecting 2,000 samples as the development set. The alpaca dataset consists of 51K instruction-following examples generated by GPT-3.5 (Wang et al., 2023) , covering a variety of tasks and question formats, and it is designed to help the language model learn how to better understand and respond to instructions. We follow INSTRUCTEVAL (Chia et al., 2024) for evaluation, employing the MMLU (Hendrycks et al., 2021), BBH (Srivastava et al., 2023), DROP (Dua et al., 2019), and HumanEval (HEval) (Chen et al., 2021) datasets.

Implementation Details. For all methods, we set the rank r to 64. For ASLoRA, the maximum number of merges is set to 16. In terms of task setting,

the MMLU uses 5-shot direct prompting, the BBH and DROP (dev) use 3-shot direct prompting, and the HEval uses 0-shot direct prompting, which reflects the complexity of different tasks and their requirements for model inference ability. During the training process, we use the AdamW optimizer, and train models for 3 epochs. The learning rate was based on a linear scheduling strategy, with an initial value of 3×10^{-4}. The batch size is set to 128. The above configuration ensures the consistency of experimental conditions and helps to comprehensively evaluate the performance of the model in each task. We provide the hyperparameters in Table 4 in Appendix.

Results. The results are shown in Table 2, we find that ASLoRA uses only 26% of the parameters required by other efficient fine-tuning methods, outperforms all baseline approaches on the BBH, DROP, and HEval datasets. While slightly underperforming full fine-tuning on the MMLU dataset, ASLoRA outperforms LoRA and its variants. Furthermore, ASLoRA achieves the highest average performance among the evaluated methods. These findings demonstrate that ASLoRA's integration of global and partial sharing mechanisms efficiently captures shared features across layers and allocates knowledge flexibly based on task demands. Consequently, ASLoRA significantly enhances model adaptability to diverse task complexities while preserving parameter efficiency, underscoring its promise in efficient fine-tuning.

4.3 Further Analyses

Advantage of Adaptive Sharing. To further investigate the advantages of adaptive sharing, we conduct a comparative experiment against fixed sharing methods. In the fixed sharing method, the B matrix is shared across every 2, 3, and 6 consecutive layers, whereas adaptive sharing merges 6, 8, and 10 times. These configurations are chosen for comparison because they maintain the same parameter counts, ensuring a fair evaluation. We conduct experiments on the MRPC, STS-B, SST-2, and QNLI datasets to evaluate the performance of adaptive sharing, with results presented in Table 3. The results indicate that adaptive sharing provides significant advantages across all configurations. For the 6-merging case (corresponding to sharing the B matrix across every 2 layers), adaptive sharing yielded the largest performance improvement. Fewer merges enable adaptive sharing to allocate knowledge more flexibly, resulting in more diverse merging outcomes. However, when the number of merges increased to 10 (corresponding to sharing the B matrix across every 6 layers), the performance advantage of adaptive sharing reduced. This is understandable, as increasing the number of merges limits the available options, reducing the diversity of adaptive sharing and making it closer to the fixed sharing method. In summary, adaptive sharing outperforms fixed sharing in terms of both parameter efficiency and performance, with its flexibility and adaptability offering significant advantages, particularly in configurations with fewer merges.

Method	#Params	MRPC	STS-B	SST-2	QNLI
ASLoRA$^{\text{-adp}}$	0.086M	88.48	90.80	94.27	92.15
ASLoRA	0.086M	**90.20**	**90.92**	**94.61**	**92.93**
ASLoRA$^{\text{-adp}}$	0.061M	88.73	**90.79**	94.50	92.75
ASLoRA	0.061M	**88.97**	90.73	**94.84**	**92.84**
ASLoRA$^{\text{-adp}}$	0.037M	**89.22**	90.42	**94.27**	92.20
ASLoRA	0.037M	**89.22**	**90.43**	**94.27**	**93.10**

Table 3: Performance on adaptive sharing and fixed sharing is compared. ASLoRA$^{\text{-adp}}$ represents fixed sharing, with results corresponding to sharing matrix B across every 2, 3, or 6 consecutive layers from top to bottom. These results are compared with adaptive sharing after merging 6, 8, and 10 times.

Shared Distribution. To explore the impact of adaptive sharing on model structure, we conduct experiments on the RoBERTa-base model and reported the results of 6 merge iterations on the MRPC and QNLI datasets (as shown in Figure 3). The results indicate that adaptive sharing achieves a more diversified allocation strategy. In the query matrix, the differences between adaptive sharing and fixed sharing are minimal, especially on the QNLI dataset, where the first 8 layers almost exclusively use adjacent two-layer sharing, with only slight differences appearing in the last four layers. This suggests that in the query matrix, inter-layer feature differences are small, and the performance of adaptive sharing and fixed sharing is similar. However, in the value matrix, the differences are more pronounced. Adaptive sharing exhibits a distinctly different sharing pattern, particularly on the QNLI dataset, where greater divergence is seen, especially in the sharing of layers 6, 8, and 9. Comparing MRPC and QNLI datasets, we find that adaptive sharing presents a more

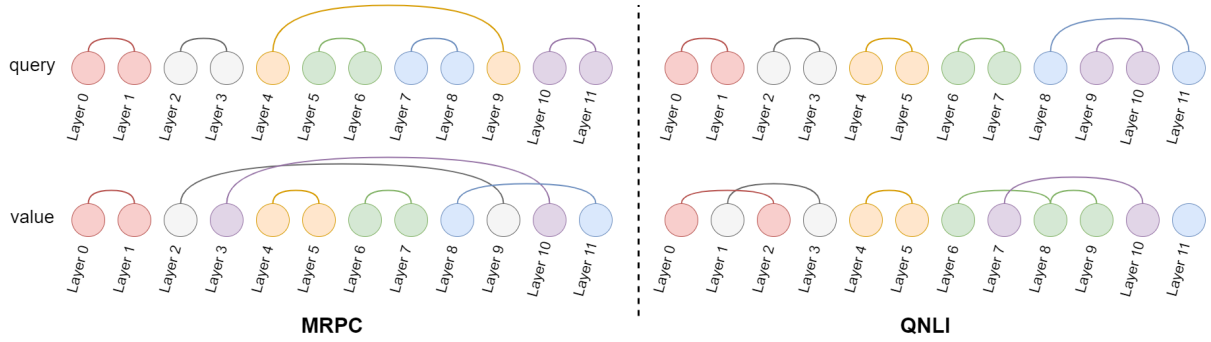

Figure 3: The allocation results of adaptive sharing on the GLUE Benchmark are presented. We set the merge times to 6 and report the sharing configurations of the query and value matrices. The same color represents sharing the same B matrix. More results can be found in Figure 5 in Appendix.

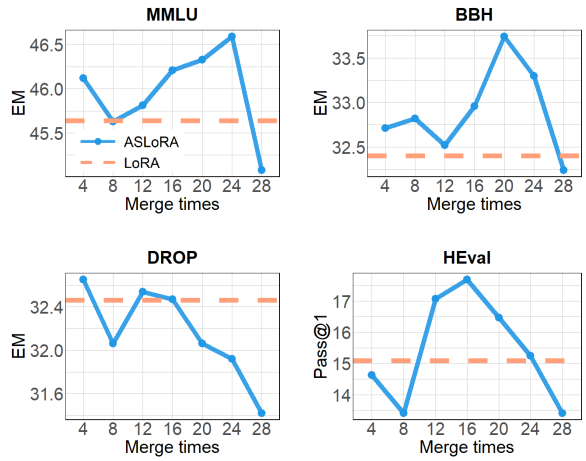

Figure 4: The effect of the number of merges on the results. Across these 4 datasets, for ASLoRA, we set the merge counts N ={4, 8, 12, 16, 20, 24, 28} and conduct a comparative analysis with LoRA under the same rank r setting.

diverse allocation pattern on the QNLI dataset. This is because the QNLI dataset is larger and more complex than the MRPC dataset, providing a richer feature structure for the model to learn. In summary, adaptive sharing can flexibly adjust the sharing strategy for each layer, significantly enhancing model performance and highlighting its advantages in model structure flexibility and task adaptability. In addition, ASLoRA can demonstrate more detailed allocation patterns on the value matrix and larger, more complex datasets.

Impact of Merge Times. We explore the impact of different merge counts on the performance of instruction tuning, setting the merge counts $N = \{4, 8, 12, 16, 20, 24, 28\}$ and comparing the results with LoRA under the same rank settings, as shown in Figure 4. The results show that on the MMLU, BBH, and HEval datasets, performance improves initially with increasing merge counts but declines beyond a certain point. In most configurations, ASLoRA outperforms LoRA. Specifically, the best performance is achieved with $N = 24$ for MMLU, $N = 20$ for BBH, and $N = 16$ for HEval. This indicates that the optimal merge count varies across datasets. The number of parameters decreases as the number of merges increases. Moderate merging helps mitigate overfitting, but excessive merging can harm performance. Conversely, fewer merges lead to an increase in the number of parameters, and too few merges may result in overfitting risks and lower parameter efficiency. On these four datasets, we also find that ASLoRA performs worse on the DROP dataset compared to the others. This may be due to the complexity of the tasks in this dataset, which makes it difficult for the reduced parameters to effectively capture its intricate features.

5 Conclusion

In this paper, we propose a parameter-efficient fine-tuning method called ASLoRA, which employs a cross-layer parameter-sharing mechanism combining global sharing and partial adaptive sharing strategies. This approach significantly enhances parameter efficiency during fine-tuning. Extensive experiments demonstrate that ASLoRA reduces the number of parameters while improving model performance across multiple datasets.

6 Limitations & Future Work

This work has the following limitations:

- We introduce two hyper-parameters: the starting merge step and the interval between merges. Different configurations of these parameters may

lead to performance variations. For the starting merge step, although we find that setting it to around an epoch yields good results, better patterns may exist. For the merge interval, we plan to introduce the global budget scheduler from AdaLoRA to design a more effective strategy for spacing between merges, thereby further optimizing performance.

- The optimal number of merges varies across datasets. In future work, we plan to integrate a dynamic search algorithm to automatically determine the optimal number of merges, enhancing the model's adaptability and overall performance.
- Our current approach is limited to inter-layer parameter sharing, which could potentially be complemented by incorporating intra-layer parameter sharing. Additionally, the method does not modify the internal structure of LoRA. In future work, our approach can be combined with other parameter-reduction methods that improve the LoRA structure (e.g., MELoRA) to achieve higher parameter efficiency.

References

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. 2023. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805.

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. 2024. Relaxed recursive transformers: Effective parameter sharing with layer-wise lora. *CoRR*, abs/2410.20672.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. *Unknown Journal*, pages 1–14.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 2778–2788. ACM.

Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. 2024. InstructEval: Towards holistic evaluation of instruction-tuned large language models. In *Proceedings of the First edition of the Workshop on the Scaling Behavior of Large Language Models (SCALE-LLM 2024)*, pages 35–64, St. Julian's, Malta. Association for Computational Linguistics.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *International Conference on Learning Representations*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019.

DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *Unknown Journal*, pages 2368–2378.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

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

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

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. 2024. Vera: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2020. Exploring versatile generative language model via parameter-efficient transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 441–459, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Xiuqing Lv, Peng Zhang, Sunzhu Li, Guobing Gan, and Yueheng Sun. 2023. LightFormer: Light-weight transformer using SVD-based weight transfer and parameter sharing. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10323–10335, Toronto, Canada. Association for Computational Linguistics.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *CoRR*, abs/2404.02948.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. *Unknown Journal*, pages 2383–2392.

Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. 2021. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4081–4090, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten de Rijke, Zhumin Chen, and Jiahuan Pei. 2024. MELoRA: Mini-ensemble low-rank adapters for parameter-efficient

fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3052–3064, Bangkok, Thailand. Association for Computational Linguistics.

Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. 2024. Tied-LoRA: Enhancing parameter efficiency of LoRA with weight tying. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8694–8705, Mexico City, Mexico. Association for Computational Linguistics.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4222–4235. Association for Computational Linguistics.

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

Yurun Song, Junchen Zhao, Ian G. Harris, and Sangeetha Abdu Jyothi. 2024. Sharelora: Parameter efficient and robust large language model fine-tuning via shared low-rank adaptation. *CoRR*, abs/2406.10785.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Trans. Mach. Learn. Res.*, 2023.

Sho Takase and Shun Kiyono. 2023. Lessons on parameter sharing across layers in transformers. In *Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing, SustaiNLP 2023, Toronto, Canada (Hybrid), July 13, 2023*, pages 78–90. Association for Computational Linguistics.

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 (2023). *URL https://github. com/tatsu-lab/stanford_alpaca*, 1(9).

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Sheng Wang, Boyang Xue, Jiacheng Ye, Jiyue Jiang, Liheng Chen, Lingpeng Kong, and Chuan Wu. 2024. PRoLoRA: Partial rotation empowers more parameter-efficient LoRA. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2829–2841, Bangkok, Thailand. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

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

Feiyu Zhang, Liangzhi Li, Junhao Chen, Zhouqiang Jiang, Bowen Wang, and Yiming Qian. 2023a. Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning. *CoRR*, abs/2308.12043.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023b. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*.

A Hyper-parameters

The detailed hyper-parameter settings on the instruction tuning and GLUE datasets are listed in Table 4 and Table 5.

Hyper-Parameter	Value
Learning rate η	3e-4
Batch size	128
Number of epochs	3
Max sequence length	256
Rank r	4
Start Steps T_s	400
Merge interval $\mathcal{W}$	10
LoRA dropout	0.05
LoRA alpha α	16
Trainable matrices	W_Q, W_V
LR scheduler	Linear
Warmup steps	100

Table 4: The hyper-parameter settings for instruction tuning. We use the same settings as (Chia et al., 2024).

Hyper-Parameter	SST-2	MRPC	CoLA	QNLI	RTE	STS-B
Learning Rate η	5e-4	4e-4	4e-4	4e-4	4e-4	4e-4
Batch Size	16	16	32	32	16	16
Number of Epochs	60	30	80	25	25	40
Weight Decay β	0.1	0.1	0.1	0.1	0.1	0.1
Max Sequence Length	512	512	512	512	512	512
Start Steps T_s	3000	320	400	1000	200	700
Merge interval $\mathcal{W}$	2000	240	500	700	100	500
Update Ratio λ	0.5	0.5	0.5	0.5	0.5	0.5
Rank r	8	8	8	8	8	8
Alpha α	16	16	16	16	16	16
LR Scheduler	Linear	Linear	Linear	Linear	Linear	Linear
Trainable Matrices	W_Q, W_V	W_Q, W_V	W_Q, W_V	W_Q, W_V	W_Q, W_V	W_Q, W_V
Warmup Ratio	0.06	0.06	0.06	0.06	0.06	0.06
Evaluation Metrics	Accuracy	Accuracy	Matthews	Accuracy	Accuracy	Pearson

Table 5: The hyper-parameter settings for GLUE.

B Details of Datasets

B.1 GLUE Benchmark

The GLUE (Wang et al., 2018) (General Language Understanding Evaluation) benchmark is a collection of natural language understanding tasks designed to evaluate the performance of language models in various practical applications. It provides a standardized platform for comparing how different models perform in understanding and processing human language. The GLUE benchmark includes nine tasks, each aiming to test different aspects of language understanding, such as text classification, sentence similarity, and reasoning. These tasks are MNLI (Williams et al., 2018)(inference), SST-2 (Socher et al., 2013) (sentiment analysis), MRPC (Dolan and Brockett, 2005) (paraphrase detection), CoLA (Warstadt et al., 2019) (linguistic acceptability), QNLI (Rajpurkar et al., 2016) (inference), QQP (question-answering),

RTE (inference), and STS-B (Cer et al., 2017) (textual similarity), we summarize their statistics in Table 6.

Corpus	Task	# Train	# Val	# Test	# Labels	Metrics	Domain
Single-Sentence Tasks							
CoLA	Acceptability	8.55k	1.04k	1.06k	2	Matthews Corr.	misc.
SST-2	Sentiment	67.3k	872	1.82k	2	Accuracy	Movie reviews
Similarity and Paraphrase Tasks							
MRPC	Paraphrase	3.67k	408	1.73k	2	Accuracy/F1	News
STS-B	Sentence similarity	5.75k	1.5k	1.38k	1	Pearson/Spearman Corr.	misc.
QQP	Paraphrase	364k	40.4k	391k	2	Accuracy/F1	Social QA
Inference Tasks							
MNLI	NLI	393k	19.65k	19.65k	3	Accuracy	misc.
QNLI	QA/NLI	105k	5.46k	5.46k	2	Accuracy	Wikipedia
RTE	NLI	2.49k	277	3k	2	Accuracy	News & Wikipedia

Table 6: Summary of GLUE benchmark tasks.

B.2 Instruction Tuning

- **MMLU** (Hendrycks et al., 2021) evaluates models' knowledge and problem-solving skills across various fields. It tests performance in zero-shot and few-shot settings, making it highly challenging and closely aligned with human evaluation standards. The dataset covers 57 subjects, including STEM, humanities, and social sciences, with difficulty levels ranging from elementary to advanced professional. Each sample provides four choice of answers, and the task is to select the correct one.
- **BBH** (Srivastava et al., 2023) is a high-difficulty subset of the BIG-Bench benchmark, comprising 23 tasks designed to test scenarios that are challenging for current language models. These tasks include complex instructions such as navigation, logical reasoning, and fallacy detection.
- **DROP** (Dua et al., 2019) is a math-focused reading comprehension benchmark that requires logical reasoning over Wikipedia-based passages. Models need to resolve references in the questions and perform discrete operations such as addition, counting, and sorting.
- **HumanEval** (Chen et al., 2021) is a benchmark for evaluating code generation models. It includes 164 original programming tasks that assess language understanding, algorithms, and basic mathematical reasoning. Some problems resemble those found in basic coding interviews.

C Analysis of Shared Distribution.

In Figure 5, we present the results of additional adaptive allocations. For all datasets, the value matrix exhibits more diverse assignment patterns, with the complexity of these assignments varying across different datasets. Among them, SST-2 shows the most detailed allocation, likely due to its larger data size and more complex task.

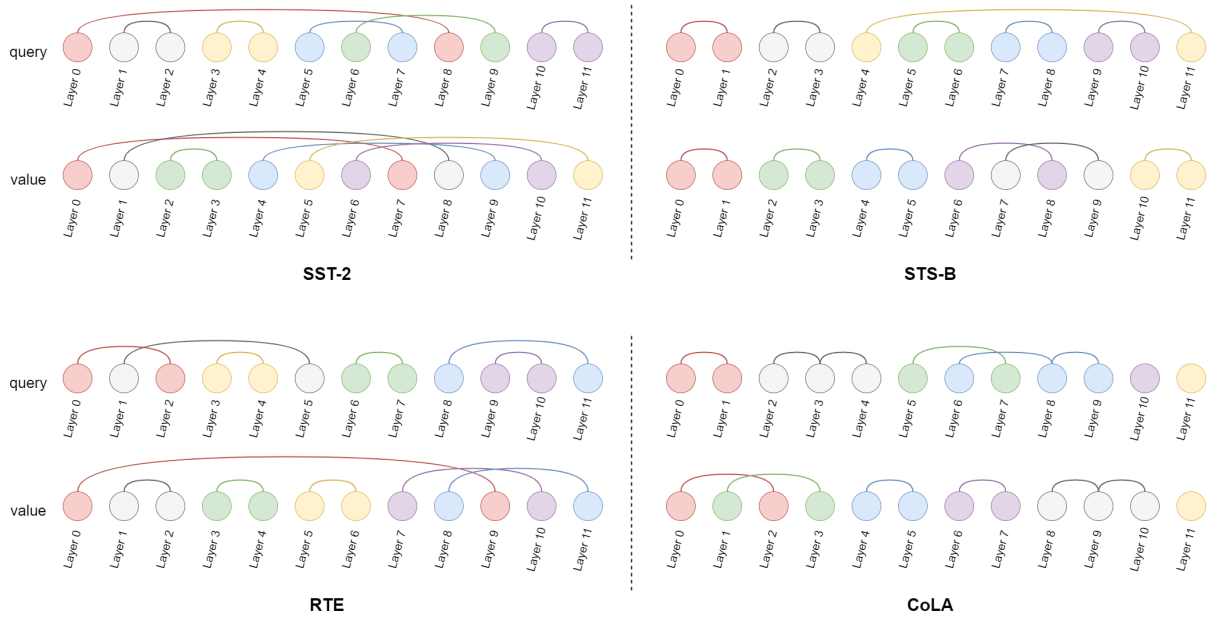

Figure 5: The allocation results of adaptive sharing on the GLUE Benchmark are presented. We set the merge times to 6 and report the sharing configurations of the query and value matrices. The same color represents sharing the same B matrix.