Rethinking Parameter Sharing for LLM Fine-Tuning with Multiple LoRAs

Hao Ban and Kaiyi Ji*
Department of Computer Science and Engineering, University at Buffalo
{haoban, kaiyiji}@buffalo.edu

Abstract

Large language models are often adapted using parameter-efficient techniques such as Low-Rank Adaptation (LoRA), formulated as $y = W_0x + BAx$, where W_0 is the pre-trained parameters and x is the input to the adapted layer. While multi-adapter extensions often employ multiple LoRAs, prior studies suggest that the inner A matrices are highly similar during training and thus suitable for sharing. We revisit this phenomenon and find that this similarity is largely attributable to the identical initialization rather than shared knowledge, with B playing a more critical role in knowledge encoding and transfer. Motivated by these insights, we propose **ALoRA**, an asymmetric multi-LoRA design with multiple A matrices and a single shared B in multi-task fine-tuning, and **Fed-ALoRA**, which shares B across clients in federated fine-tuning under both homogeneous and heterogeneous settings, through a novel matrix decomposition strategy to accommodate heterogeneous ranks across clients. Experiments on commonsense reasoning, math reasoning, multi-task NLP dataset, and federated NLP dataset demonstrate that our methods achieve more balanced performance across tasks with comparable or superior average accuracy relative to existing multi-LoRA approaches. The code is available at https://github.com/OptMN-Lab/ALoRA.

1 Introduction

Large language models (LLMs) have achieved remarkable performance across diverse domains (Comanici et al., 2025; Dubey et al., 2024; Achiam et al., 2023), but the growing scale makes conventional full fine-tuning increasingly expensive. Parameter-efficient fine-tuning (PEFT) addresses this challenge by freezing the pre-trained model and updating only a small subset of parameters, improving efficiency while maintaining performance (Han et al., 2024). Among PEFT methods, Low-Rank Adaptation (LoRA) (Hu et al., 2022) is particularly popular: it decomposes weight updates into trainable low-rank matrices A and B, which can be merged into the pre-trained model without extra inference latency.

Recent studies have shown that a single LoRA has limited capacity when handling diverse data distributions (Cai et al., 2025; Yang et al., 2024). A natural extension is to use multiple LoRAs, where each module can specialize in different data modes such as tasks, domains, and distributed clients (Liao et al., 2025; Sun et al., 2025; Li et al., 2024; Wu et al., 2024b). In multi-task fine-tuning, adapters are required to handle task heterogeneity (Liang et al., 2025), and in federated fine-tuning, they should account for client heterogeneity and personalization (Bian et al., 2025). However, naively employing multiple LoRAs also increases computation and communication costs, which makes this approach less efficient.

To address this problem, recent methods explore parameter sharing across LoRA modules to improve parameter efficiency. HydraLoRA (Tian et al., 2024) observes that A matrices trained on different tasks exhibit very high similarity, and proposes a single shared A with multiple Bs for multi-task fine-tuning. FedSA-LoRA (Guo et al., 2025) reports similar findings in federated fine-tuning and transmits only A matrices for server aggregation with reduced communication costs. These studies attribute the high similarity in A matrices to the shared knowledge.

In this paper, we revisit this similarity phenomenon and find that the similarity of A stems mainly from identical initialization rather than shared knowledge. Our analysis of the learning dynamics suggests that A tends to act as a feature projector, whereas B captures domain specific knowledge. Empirically, we observe that sharing B often leads to more effective knowledge transfer

*Corresponding Author

Findings of the Association for Computational Linguistics: ACL 2026, pages 12843–12863

than sharing A in both multi-task and federated fine-tuning settings. These insights motivate an interesting but underexplored question: *might sharing the module B, rather than A, be more effective for parameter and knowledge sharing?* In this paper, we provide a positive answer, with our main contributions as follows.

- We propose **ALoRA**, an asymmetric multi-LoRA architecture for multi-task fine-tuning. It employs multiple A matrices and a single shared B matrix, where the A matrices are dynamically routed by the inputs. This design enables each A to explore distinct feature subspaces while encouraging knowledge transfer through the shared B.
- We propose **Fed-ALoRA**, which communicates only B matrices rather than full LoRA parameters for aggregation on server. It supports both homogeneous and heterogeneous settings with the same and different ranks across clients, whereas existing parameter-sharing federated fine-tuning methods focus only on the homogeneous case. In the homogeneous setting, Fed-ALoRA updates all A matrices locally, and transmits and aggregates only B matrices on server side. In the heterogeneous setting, direct aggregation of B is infeasible due to their distinct sizes, so we decompose B into (B_1, B_2) with appropriate sizes and introduce an auxiliary matrix for further dimension adjustment. Compared to full LoRA aggregation, Fed-ALoRA reduces communicated parameters by up to 50% and 75% in the homogeneous and heterogeneous settings, respectively, while maintaining performance.
- We conduct extensive experiments on intra-domain multi-task benchmarks such as commonsense and math reasoning, cross-domain multi-task NLP dataset, and federated NLP dataset, using major open-source models such as LLaMA and Qwen. Across all datasets, our methods consistently deliver more balanced performance with comparable or superior accuracy compared to existing methods. In particular, **ALoRA** surpasses the sharing-A approach **HydraLoRA** on LLaMA2-7B, improving average ROUGE-1 by +0.68 with a $\Delta m\%$ (which quantifies performance balance via mean drop from single-objective baselines) gain of -1.94. Similarly, **Fed-ALoRA** outperforms the sharing-A approach **FedSA-LoRA**, achieving gains of +1.26 (homogeneous) and +1.96 (heterogeneous) with $\Delta m\%$ gains of -2.08 and -2.65, respectively. Similar improvements can also be observed on Qwen2-7B. Compared with approaches that aggregate full LoRA parameters, our method attains comparable performance, smaller $\Delta m\%$, and substantially reduced communication cost by transmitting much fewer parameters.

2 Background

2.1 Low-Rank Adaptation

Pre-trained language models exhibit low intrinsic dimensionality when adapted to downstream tasks (Aghajanyan et al., 2021). LoRA leverages this property by approximating weight updates through low-rank decomposition. Particularly, for a pre-trained weight matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the weight updates is defined as $\Delta W = BA$, where $A \in \mathbb{R}^{r \times d_{\text{in}}}$, $B \in \mathbb{R}^{d_{\text{out}} \times r}$, and the rank $r \ll min(d_{\text{in}}, d_{\text{out}})$. During training, only A and B matrices are trainable. In practice, A is typically initialized using Kaiming Uniform (He et al., 2015), and B is initialized as zero.

2.2 Fine-Tuning with Multiple LoRAs

Multiple LoRA-based methods extend vanilla LoRA with additional modules to improve adaptability across heterogeneous domains (Dettmers et al., 2023; Zi et al., 2023). In multi-task fine-tuning, they often use MoE designs where LoRAs act as dynamically routed experts (Huang et al., 2024; Luo et al., 2024), while in federated fine-tuning, they aim to balance personalization with shared knowledge aggregation (Raje et al., 2025; Zhang et al., 2025b). A common idea among these multi-LoRA approaches is to share the same matrix A, based on the observation that A matrices from LoRAs trained on different tasks or clients are often highly similar. For example, HydraLoRA (Tian et al., 2024) employs a single A matrix and multiple B matrices to express the weight updates: $\Delta W = \sum_{i=1}^{n} w_i B_i A$, where n is the number of B matrices, w_i is the gating score for each B_i. The federated multi-LoRA approach FedSA-LoRA (Guo et al., 2025) shares only the A matrices for server aggregation, after which the server broadcasts the aggregated A to all clients. The model update of client i is given by: $\Delta W_i^t = B_i^t \bar{A}^t$, and $\bar{A}^t = \text{Agg}(A_1^{t-1}, \cdots, A_n^{t-1})$, where n is the number of clients, and t is the current communication round, and $\text{Agg}(\cdot)$ denotes an aggregation algorithm such as simple averaging.

3 Revisiting Parameter Sharing in Multi-LoRA Fine-tuning

As noted earlier, a common strategy for parameter sharing is to reuse the same matrix A across multiple LoRA modules, with the goal of reducing the total number of parameters and enabling knowledge transfer. In this section, we systematically re-examine this approach through a series of controlled experiments. Full implementation details and similar results on Qwen2 model are provided in Appendix B.

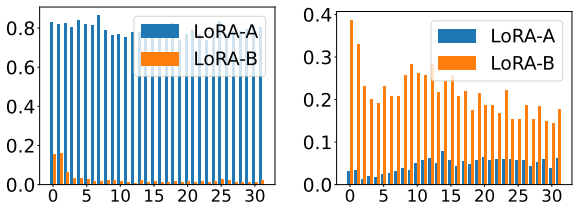
3.1 Similarity in A Stems from Same Initialization, Not Shared Knowledge

A primary motivation for sharing A across LoRA modules is the observation that the matrices A_i of different LoRAs often appear similar during training. However, upon closer examination, we find that this similarity largely arises from their common initialization rather than from the shared knowledge. We fine-tune the LLaMA2-7B model (Touvron et al., 2023) separately on classification and summarization tasks from the Dolly-15K dataset (Conover et al., 2023), using either identical or different random seeds for A initialization (leading to different initializations for the matrices A_i), and compare the resulting LoRA modules using principal angle-based similarity (Zhu and Knyazev, 2013), where a value of 1 indicates complete similarity and 0 indicates dissimilarity. The results are shown in Figure 1a-1c, and details of the similarity metric are discussed in Appendix B.1.

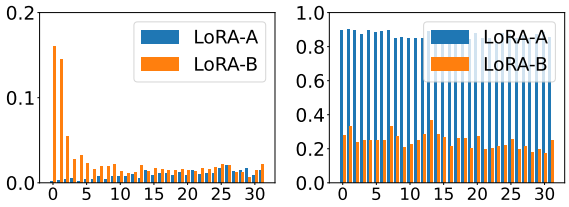
Observation. Figure 1a shows that with the same initialization, A matrices are highly similar. In contrast, Figure 1b-1c show that with different initializations, A matrices from either the same or different tasks exhibit little similarity, while B matrices display relatively higher similarity. These results suggest that the A is highly sensitive to random seeds rather than necessarily capturing shared knowledge, whereas B is less affected.

3.2 Dissecting Distinct Dynamics of A and B During Training

The above analysis motivates us to further investigate the learning dynamics of A and B during training by comparing their states before and after fine-tuning on the summarization task[1]. Our



(a) Different tasks, same init. (b) Same task, different init.



(c) Different tasks and init. (d) Before and after FT.

Figure 1: Layer-wise similarity analysis. The x-axis denotes the layer indices, and the y-axis denotes the similarity scores. Subfigures (a)-(c) compare different LoRA modules under varying tasks and initialization settings: (a) two different tasks with the same random seed; (b) the same task with different random seeds; and(c) two different tasks with different random seeds. Subfigure (d) compares the same LoRA module before and after fine-tuning. A matrices are similar only under the same initialization, whereas B exhibits relatively stable similarity across different tasks and seeds. In addition, A remains largely unchanged from initialization.

experiments evaluate (i) the similarity of modules A and B (using the similarity metric in Section 3.1) and (ii) the magnitude and directional variations of ΔW, A and B. To formalize this[2], any weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ can be decomposed into a magnitude and a direction component: $W = \|W\|_c \frac{W}{\|W\|_c} = mV$, where $\|\cdot\|_c$ denotes the column-wise norm. Here, $m \in \mathbb{R}^{1 \times d_{\text{in}}}$ is the magnitude vector, with m_j denoting the norm of the j-th column of W, and $V \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is the direction matrix with unit-norm columns. Given two matrices W_1 and W_2, their magnitude and direction discrepancies are defined as

$$\Delta M = \frac{1}{d_{\text{in}}} \sum_{j=1}^{d_{\text{in}}} |m_{1,j} - m_{2,j}|$$

$$\Delta D = \frac{1}{d_{\text{in}}} \sum_{j=1}^{d_{\text{in}}} (1 - \cos(V_{1,j}, V_{2,j})).$$

Observation. Figure 1d shows that A remains highly similar throughout training, undergoing only minimal changes, whereas B exhibits much larger differences, indicating substantial adaptation after fine-tuning. Figure 2 further reveals that the varia-

[1] We use the checkpoints from the second and final steps, since B is initialized to zero at the beginning.

[2] We follow the same setup as in Liu et al. (2024).

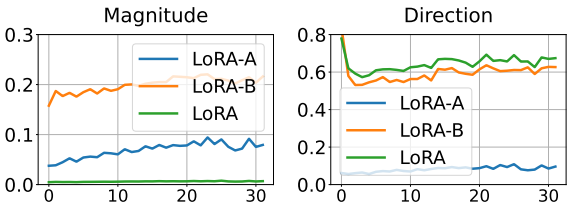


Figure 2: Comparison of LoRA modules variations before and after the fine-tuning. The x-axis denotes the layer indices, and the y-axis denotes the variation values. The module B exhibits pronounced variation in both magnitude and direction. Overall, LoRA shows limited magnitude change, with nearly all directional change captured by B.

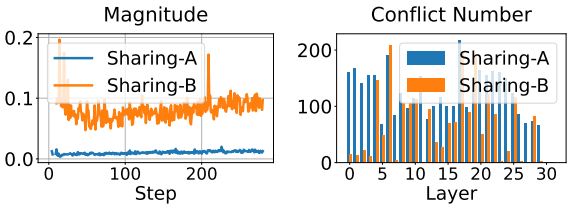


Figure 3: Comparing sharing A versus B in multi-task fine-tuning. Left: gradient magnitudes of A and B. Right: number of gradient conflicts per layer. Sharing A causes smaller gradient magnitudes and more frequent conflicts than sharing B.

tions in A are primarily in magnitude with little directional change, while B accounts for most of the direction change. We also conduct variation analysis using different models and settings and observe similar findings. The details are provided in Appendix B.2. These results suggest that A functions more as a fixed feature projector, whereas B aggregates and adapts these features to encode domain knowledge. This highlights the more dominant role of B over A in knowledge learning, raising an intriguing question: might sharing the module B, rather than A, be more effective for parameter and knowledge sharing?

3.3 Sharing A or Sharing B?

In this section, we address the question from Section 3.2 by comparing the performance of sharing modules A and B.

Gradient conflicts lead to lazy learning for A in multi-task fine-tuning. Given an input $x \in \mathbb{R}^{d_{\text{in}}}$ and the output gradient $g \in \mathbb{R}^{d_{\text{out}}}$, the gradient of A in sharing-A structure is $\nabla A = \sum_{i=1}^{n} w_i (B_i^\top g) x^\top$, where each term corresponds to a B_i expert. We record the magnitudes of ∇A and compute cosine similarity between gradient components, where negative similarity indicates a conflict that may hinder learning. We apply same procedure to the sharing-B structure and compare them on commonsense reasoning (Hu et al., 2023). We track gradient magnitudes of shared parameters and the number of conflicts during training.

Observation. Figure 3(Left) shows that the gradient magnitude of A in the sharing-A structure is near zero, while the gradient of B in the sharing-B structure is much larger. Figure 3(Right) shows that sharing A also produces more gradient conflicts. Thus, in the sharing-A structure, A learns very slowly possibly due to the more frequent conflicting updates. We refer to this phenomenon as "lazy learning". Previous analysis in Section 3.2 indicates that A functions as a feature projector. Hence, "lazy learning" may restrict the ability to explore diverse feature subspaces.

Knowledge transfer in federated fine-tuning. Each client fine-tunes its own LoRA and transmits the shared parameters to the server, which aggregates and returns them (see Section 4.2). We assess whether the shared parameters improve knowledge transfer by evaluating each client's performance across all tasks. We compare these two structures across 8 clients, each assigned an NLP task from the FLAN dataset (Wei et al., 2022), and use ROUGE-1 score (Lin, 2004) to measure performance, where a value of 0 means no overlap between model prediction and ground truth, and 100 indicates perfect word-level overlap. We report the average score across clients in Table 1. See Appendix C.4 for detailed results for each client.

Setting	Sharing	Avg.
Homo.	A	44.30
	B	**66.32**
Hetero.	A	40.76
	B	**50.30**

Table 1: Comparing sharing A and sharing B in federated fine-tuning.

Observation. In homogeneous setting, sharing B outperforms sharing A by an average of 49.71%. In heterogeneous setting, sharing B achieves an average improvement of 23.41%. These results clearly indicate that sharing B better facilitates cross-client knowledge transfer than sharing A.

4 Proposed Methods

Motivated by the findings in Section 3, we replace A with B as the shared parameter and propose two simple and effective multi-LoRA fine-tuning methods: **ALoRA** (Asymmetric LoRA) for multi-task training, and **Fed-ALoRA** for both homogeneous and heterogeneous federated settings.

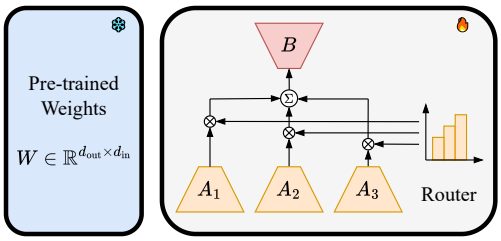


Figure 4: ALoRA adopts multiple A and a single B to explore diverse feature subspaces.

4.1 ALoRA for Multi-Task Fine-Tuning

Multi-task fine-tuning typically adapts a pretrained LLM using data from multiple tasks. The goal is to improve generalization by learning from diverse inputs. The proposed ALoRA is illustrated in Figure 4. Given input $x \in \mathbb{R}^{d_{\text{in}}}$, the forward pass is

$$y = y_0 + \Delta y = W_0 x + B \sum_{i=1}^{n} w_i A_i x,$$

where $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is the pre-trained weight matrix, $A_i \in \mathbb{R}^{r \times d_{\text{in}}}$ are the expert matrices, $B \in \mathbb{R}^{d_{\text{out}} \times r}$ is the shared aggregator, and the rank $r \ll min(d_{\text{in}}, d_{\text{out}})$. Each A_i projects the input into a distinct feature subspace, and B fuses the learned features to produce the output. The expert weights $w = (w_1, \ldots, w_n)$ are obtained from an input-aware router, implemented as a linear gating function with parameters $W_g \in \mathbb{R}^{n \times d_{\text{in}}}$: $w = \text{softmax}(W_g x)$. During the inference, the router computes input-dependent weights, and the weighted average of the adapters is dynamically merged into the pre-trained weights.

4.2 Fed-ALoRA for Federated Fine-Tuning

Federated fine-tuning can be divided into two settings: (i) *homogeneous*, where all clients adopt the same configuration, and (ii) *heterogeneous*, where clients have varying capacities, introducing both computational and communication heterogeneity.

Homogeneous setting. In this case, all n clients fine-tune their LoRA modules with the same rank. Each update takes the form $\Delta W_i = B_i A_i$, where $A_i \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B_i \in \mathbb{R}^{d_{\text{out}} \times r}$. Figure 5(Left) illustrates the procedure of Fed-ALoRA for homogeneous setting, with details for each communication round t below ($t \geq 1$):

S1: *Initialization.* If $t = 1$, each client initializes A_i randomly and sets B_i to zero. For $t > 1$, A_i and B_i are initialized with A_i^{t-1} and B_0^{t-1}.

S2: *Local training.* Each client performs LoRA fine-tuning on its local data, obtaining (A_i^t, B_i^t) by optimizing $\mathcal{L}(W_0 + B_i A_i)$ with respect to (A_i, B_i), where $\mathcal{L}(\cdot)$ is the loss function. The client then uploads only B_i^t to the server for aggregation.

S3: *Aggregation.* The server aggregates the uploaded matrices using the operator $\text{Agg}(\cdot)$ from McMahan et al. (2017), and obtains $B_0^t \leftarrow \text{Agg}(B_1^t, \cdots, B_n^t)$.

S4: *Broadcast.* The server then sends the global matrix B_0^t back to all clients.

In full-LoRA aggregation, each client communicates $(d_{\text{in}} + d_{\text{out}})r$ parameters per round. In contrast, Fed-ALoRA requires transmitting only B_i, reducing communication cost to $d_{\text{out}} r$.

Heterogeneous setting. In this case, clients may have different capacity constraints, resulting in parameterizations with diverse ranks r_i, given by $\Delta W_i = B_i A_i$, where $A_i \in \mathbb{R}^{r_i \times d_{\text{in}}}$ and $B_i \in \mathbb{R}^{d_{\text{out}} \times r_i}$ for $i = 1, \ldots, n$. Since the ranks r_i differ across clients, direct averaging of the B_i matrices is infeasible, and the aggregation strategy used in the homogeneous setting cannot be applied.

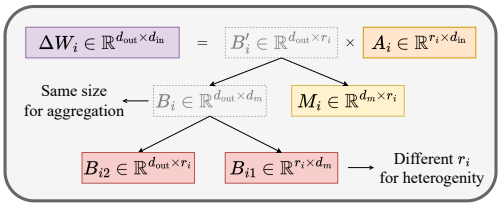


Figure 6: Our proposed decomposition strategy. All clients have the same B_i dimensionality for aggregation, and different ranks r_i to support heterogeneity.

To address this issue, we propose a novel *decomposition strategy* of the form:

$$\Delta W_i = B_{i2} B_{i1} M_i A_i,$$

where $A_i \in \mathbb{R}^{r_i \times d_{\text{in}}}$, $M_i \in \mathbb{R}^{d_m \times r_i}$, $B_{i1} \in \mathbb{R}^{r_i \times d_m}$ and $B_{i2} \in \mathbb{R}^{d_{\text{out}} \times r_i}$. The *high-level idea* is illustrated in Figure 6. We first represent the low-rank updates as $B_i M_i A_i$ for i-th client, where M_i is an intermediate matrix controlling d_m. This design enables us to aggregate B_i across all clients. To handle the heterogeneity, we further decompose B_i into two components, (B_{i1}, B_{i2}) with rank r_i. In addition, every client maintains an accumulator $B_{i0} \in \mathbb{R}^{d_{\text{out}} \times d_m}$ which stores the global updates it has received so far. The full procedure of round t is illustrated in Figure 5(Right) and detailed below:

S1: *Initialization.* If $t = 1$, each client initializes (A_i, M_i, B_{i1}) randomly, and sets (B_{i0}, B_{i2}) to

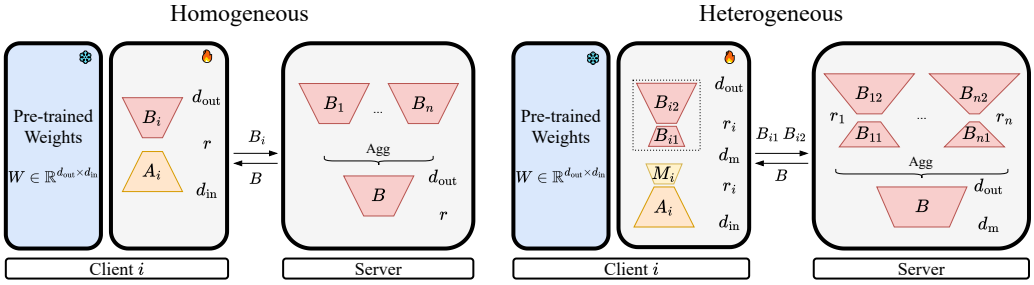


Figure 5: Fed-ALoRA shares only B matrices for server aggregation. Left: Homogeneous setting (same rank), where the shared B is directly transmitted. Right: Heterogeneous setting (different ranks), where the shared B is decomposed into two matrices for heterogeneity. Compared to the standard full LoRA aggregation, the communication cost per client is reduced to $\mathcal{O}(d_{\text{out}}r)$ in the homogeneous setting and $\mathcal{O}(d_{\text{out}}r_i)$ in the heterogeneous setting if d_m is chosen appropriately.

zero. For $t > 1$, (A_i, M_i) is initialized with (A_i^{t-1}, M_i^{t-1}), B_{i0} is updated by accumulation as $B_{i0} \leftarrow B_{i0} + B_0^{t-1}$, B_{i1} is re-initialized randomly, and B_{i2} is resets to zero.

S2: *Local training.* Each client performs LoRA fine-tuning on its local data, and obtains parameters $(A_i^t, M_i^t, B_{i1}^t, B_{i2}^t)$ by optimizing $\mathcal{L}(W_0 + (B_{i0} + B_{i2}B_{i1})M_iA_i)$ with respect to $(A_i, M_i, B_{i1}, B_{i2})$. The client then uploads parameters (B_{i1}^t, B_{i2}^t) to the server.

S3: *Aggregation.* The server reconstructs $B_i^t = B_{i2}^t B_{i1}^t$ for each client and then performs the aggregation $B_0^t \leftarrow \text{Agg}(B_1^t, \cdots, B_n^t)$.

S4: *Broadcast.* The server then sends the global matrix B_0^t back to all clients.

Remark 1. The prior parameter-sharing approach, FedSA-LoRA (Guo et al., 2025), does not support the heterogeneous setting. Fed-ALoRA addresses this limitation by introducing the decomposition (B_{i1}, B_{i2}), enabling efficient aggregation across clients with different capacities.

Remark 2. In full-LoRA aggregation, each client uploads $(d_{\text{in}} + d_{\text{out}})r_i$ parameters to the server. The server then extends all heterogeneous updates to the maximum rank $r_{\max} = \max\{r_1, \cdots, r_n\}$ by padding with zeros, and broadcasts $(d_{\text{in}} + d_{\text{out}})r_{\max}$ parameters back to every client. In Fed-ALoRA, if d_m is chosen comparable to $r_{\max}$ with $d_m \ll \min(d_{\text{in}}, d_{\text{out}})$, then client i maintains $(d_{\text{in}} + d_{\text{out}} + 2d_m)r_i \approx (d_{\text{in}} + d_{\text{out}})r_i$ trainable parameters. The communication cost is $\mathcal{O}(d_{\text{out}}r_i)$ for uplink and $\mathcal{O}(d_{\text{out}}r_{\max})$ for downlink.

5 Experiments

5.1 Multi-Task Fine-Tuning

Models and datasets. We fine-tune LLaMA3-8B (Dubey et al., 2024) and Qwen2-7B (Yang et al., 2025a) on the *intra-domain* multi-task benchmark commonsense reasoning (Hu et al., 2023), which contains 8 question answering (QA) datasets, each focusing on a different aspect of commonsense. We also fine-tune LLaMA2-7B and Qwen2-7B on the *cross-domain* multi-task NLP dataset (Long et al., 2024), which mixes 8 different tasks such as text generalization. These tasks are sampled from the FLAN dataset (Wei et al., 2022). Additional results on the math reasoning benchmark (Hu et al., 2023) are provided in Appendix C.

Baseline methods. For each task (or client in the federated setting), we first fine-tune LoRA on its own dataset and use the performance as the single-task baseline, denoted as ST Baseline. We compare our ALoRA with representative methods, including vanilla LoRA (Hu et al., 2022), LoHa (Yeh et al., 2023), AdaLoRA (Zhang et al., 2023a), MoSLoRA (Wu et al., 2024a), and HydraLoRA (Tian et al., 2024). Although our focus is on parameter sharing in a multi-LoRA structure without relying on task specific information, such as task specific transformation matrices between the LoRA A and B matrices or task specific LoRA modules, we still compare our method with multi-task fine-tuning approaches that explicitly exploit such information, including LoRAMoE (Dou et al., 2023), MTL-LoRA (Yang et al., 2025b), and CoLA (Zhou et al., 2025). Our ALoRA performs better than or comparably to theses methods. Please refer to Appendix C.10 for more details.

Evaluation metrics. To evaluate performance, we use the following metrics: (1) average accuracy for commonsense reasoning and average ROUGE-1 score for multi-task NLP dataset; and (2) $\Delta m\%$ (Maninis et al., 2019), the average per-task performance change against the single-task baseline.

Method	LLaMA3-8B		Qwen2-7B	
	Avg.	$\Delta m\%(\downarrow)$	Avg.	$\Delta m\%(\downarrow)$
ST Baseline	84.90	–	87.27	–
LoRA	83.64	1.48	85.96	1.43
LoHa	83.73	1.36	86.34	1.06
AdaLoRA	83.88	1.17	86.14	1.30
MoSLoRA	84.23	0.76	86.30	1.09
HydraLoRA	84.57	0.32	86.09	1.32
ALoRA (Ours)	**84.81**	**0.04**	**86.47**	**0.91**

Table 2: Results on intra-domain multi-task commonsense reasoning benchmark. $\Delta m\%$ measures performance balance across tasks. $\downarrow$ denotes that lower values are better. All methods use the same number of adapter parameters. We run each experiment 3 times and report the average. Full details, including per-task performance and standard deviations, are provided in Appendix C.5.

Method	LLaMA2-7B		Qwen2-7B	
	Avg.	$\Delta m\%(\downarrow)$	Avg.	$\Delta m\%(\downarrow)$
ST Baseline	63.36	–	76.16	–
LoRA	61.67	0.31	79.60	-6.78
LoHa	65.70	-5.76	78.35	-5.27
AdaLoRA	61.96	-0.41	77.61	-4.33
MoSLoRA	66.18	-6.58	79.34	-6.59
HydraLoRA	66.45	-6.39	80.03	-7.14
ALoRA (Ours)	**67.13**	**-8.33**	**80.46**	**-7.98**

Table 3: Results on cross-domain multi-task NLP datasets. $\Delta m\%$ measures performance. Full details on per-task performance are provided in Appendix C.6.

$\Delta m\% = \frac{1}{K}\sum_{k=1}^{K}(-1)^{\delta_k}(M_k - M_0)/M_0 \times 100$, where M_k is the performance of k-th task under the compared method, M_0 is the baseline performance. $\delta_k = 1$ if higher values indicate better performance, otherwise $\delta_k = 0$. This metric evaluates how well performance is balanced across multiple tasks.

Results. The results are presented in Tables 2 and 3. ALoRA achieves better average scores than existing LoRA variants with the most balanced results in both benchmarks. These results suggest that ALoRA encourages knowledge transfer.

5.2 Federated Fine-Tuning

Models, datasets and metrics. We fine-tune models LLaMA2-7B and Qwen2-7B, and evaluate on the federated datasets constructed by Long et al. (2024), which includes 8 NLP tasks from FLAN dataset (Wei et al., 2022), with each client assigned to one task. We use ROUGE-1 score (Lin, 2004) as the evaluation metric. We also report the average number of parameters (in millions) communicated per client in each round, including both uploads to the server and downloads from the server.

Method	LLaMA2-7B			Qwen2-7B		
	Avg.	$\Delta m\%(\downarrow)$	Params.	Avg.	$\Delta m\%(\downarrow)$	Params.
ST Baseline	79.67	–	–	82.75	–	–
FedIT	82.47	-3.92	8.39	82.80	-0.05	6.42
FedDPA	81.96	-3.30	16.78	83.17	-0.60	12.85
FedSA-LoRA	81.15	-2.21	**4.19**	83.69	-1.15	**3.21**
Fed-ALoRA (Ours)	**82.51**	**-4.29**	**4.19**	**84.30**	**-2.05**	**3.21**

Table 4: Results for the **homogeneous** federated setting. Params. denotes the average number of parameters (in millions) transmitted per client in each round. ALoRA achieves the most balanced performance while reducing communication cost by 50% compared to full LoRA aggregation FedIT. Full details on per-client performance are provided in Appendix C.7.

Method	LLaMA2-7B			Qwen2-7B		
	Avg.	$\Delta m\%(\downarrow)$	Params.	Avg.	$\Delta m\%(\downarrow)$	Params.
ST Baseline	81.73	–	–	83.43	–	–
ZeroPadding	82.29	-0.91	49.28	83.32	0.06	37.73
FLoRA	80.45	1.54	141.56	81.65	2.13	108.38
Fed-ALoRA (Ours)	**82.50**	**-1.07**	**12.12**	**84.13**	**-1.02**	**9.23**

Table 5: Results for the **heterogeneous** setting. ALoRA achieves the most balanced performance while reducing communication cost by 75% compared to full LoRA aggregation ZeroPadding. FedSA-LoRA is not included here because it does not support heterogeneity. Full details on per-client performance are provided in Appendix C.8.

Baseline methods. We compare our Fed-ALoRA with state-of-the-art federated fine-tuning approaches including FedIT (Zhang et al., 2024), FedDPA (Long et al., 2024), FedSA-LoRA (Guo et al., 2025), ZeroPadding (Wang et al., 2024), and FLoRA (Wang et al., 2024). In the homogeneous setting, all clients use rank 8. In the heterogeneous setting, the ranks are $\{64, 64, 32, 32, 16, 16, 8, 8\}$, with d_m set to 16. Full details are provided in the Appendix C.

Results. The results are presented in Tables 4 and 5. Fed-ALoRA achieves the highest average score and most balanced performance while reducing communication cost by 50% compared to full-LoRA aggregation in homogeneous setting, and reducing by 75% in heterogeneous setting. These results show that Fed-ALoRA effectively promotes knowledge sharing across clients.

5.3 In-Depth Analysis

Multi-task fine-tuning. We analyze gate activations of HydraLoRA and ALoRA during inference on the commonsense reasoning using LLaMA3-8B. Figure 7 presents the t-SNE visualization in the last layer. ALoRA forms clearer clusters than HydraLoRA. Some tasks share similar gate activations, suggesting that they prefer same A experts.

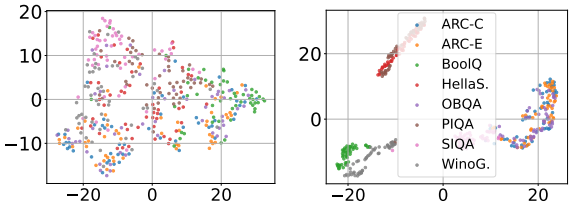


Figure 7: Gate activations of HydraLoRA (Left) and ALoRA (Right). ALoRA yields more distinct clusters.

This indicates that ALoRA captures diverse feature subspaces more effectively. In contrast, HydraLoRA relies on a single A matrix, which limits diversity and leads to more scattered activations.

We also compare HydraLoRA and ALoRA on commonsense reasoning tasks using Qwen2.5-14B (Yang et al., 2025a) to further validate our method. ALoRA achieves an average accuracy of 89.59, while HydraLoRA achieves 89.50. Full details are provided in Appendix C.11. These results show that sharing B consistently outperforms sharing A as the model size increases.

Federated fine-tuning. We study the effect of different choices of the intermediate rank d_m in the heterogeneous setting using LLaMA2-7B model. The results in Table 6 show that with a proper choice of the rank d_m, we can reduce the communication cost while maintaining the performance balance.

Fed-ALoRA	Avg.	$\Delta m\%(\downarrow)$
$d_m = 8$	81.92	-0.41
$d_m = 16$	**82.50**	**-1.07**
$d_m = 32$	82.25	-0.80
$d_m = 64$	82.37	-1.01

Table 6: Results of different intermediate ranks in the heterogeneous setting.

6 Related Work

Low-rank adaptation. Vanilla LoRA (Hu et al., 2022) reparameterizes weight updates using low-rank matrices, enabling efficient fine-tuning without extra inference latency. Extensions develop along three directions. For rank allocation, AdaLoRA (Zhang et al., 2023a) prunes less important singular values, and DyLoRA (Valipour et al., 2023) trains LoRA blocks with different ranks for flexible inference. For memory efficiency, QLoRA (Dettmers et al., 2023) applies 4-bit quantization, and SparseLoRA (Khaki et al., 2025) updates only a sparse subset of parameters using SVD. For structural variation, LoHa and LoKr (Yeh et al., 2023) adopt Hadamard and Kronecker decompositions, and DoRA (Liu et al., 2024) separates magnitude and direction. This paper provides a deep investigation into the training dynamics of modules A and B, demonstrating the more dominant role of B in knowledge learning and transfer.

Multi-task fine-tuning. Fine-tuning on multiple tasks improves generalization and transfer. A popular idea is to integrate LoRA with MoE, where experts specialize in different tasks. Among them, LoRAMoE (Dou et al., 2024), MoELoRA (Luo et al., 2024) and MoRE (Zhang et al., 2025a) align experts with task information to balance performance. SMoRA (Zhao et al., 2025) treats each rank as an expert, and ThanoRA (Liang et al., 2025) builds task-aware LoRA modules. DynMoLE (Li et al., 2025a) uses entropy-based routing, and HydraLoRA (Tian et al., 2024) improves parameter efficiency by sharing A matrices. In contrast to HydraLoRA, our proposed **ALoRA** shares matrices B, which promotes diverse feature projections and facilitates more effective knowledge transfer.

Federated fine-tuning. Models are adapted across clients while preserving data privacy. Existing methods fall into homogeneous and heterogeneous settings. In homogeneous case, FedIT (Zhang et al., 2024) aggregates full LoRA parameters, while FedSA-LoRA (Guo et al., 2025) reduces communication by sharing only A matrices. In heterogeneous case, HetLoRA (Cho et al., 2024) supports varying ranks via self-pruning with sparse aggregation, Ravan (Raje et al., 2025) introduces multi-head LoRA updates, and FedALT (Bian et al., 2025) employs MoE-based adapters; FLoRA (Wang et al., 2024) provides a unified stacking framework.

As we were preparing the final draft of this paper, we became aware of a concurrent work, MASA (Dong et al., 2025), which was posted on arXiv around the same time. MASA also explores a structure with multiple A matrices and a single B matrix. However, there are several key differences between MASA and our method. (i) *Motivation*. MASA is motivated by the observation that using a single A matrix can create an information bottleneck, whereas our work studies which parameters should be shared to facilitate effective knowledge transfer. (ii) *Architecture design*. MASA shares multiple A matrices across adjacent layers, while we adopt a simpler design in which each layer has its own adapted modules. (iii) *Application*. While both methods address multi task fine tuning, our approach is further extended to federated fine tuning with substantial modifications and significantly reduced communication.

7 Conclusion

Our study shows that the similarity of LoRAs' A matrices arises mainly from initialization rather than shared knowledge, with B serving as the key component for knowledge transfer. Building on this insight, we propose ALoRA and Fed-ALoRA, which share B for multi-task and federated fine-tuning. Experiments across diverse benchmarks demonstrate that these methods achieve more balanced performance while maintaining or improving accuracy over existing multi-LoRA approaches. Future work will further examine the distinct learning dynamics of A and B and develop new fine-tuning strategies inspired by these insights.

Limitations

In this paper, we primarily focus on multi-task and federated fine-tuning settings. Our ALoRA framework adopts a simple mixture-of-experts (MoE) structure; while effective, this design choice is not exhaustive, and other more advanced and complex MoE architectures or parameter-routing mechanisms may further improve performance or efficiency. Moreover, although our experiments cover representative language modeling tasks, some important scenarios remain unexplored, such as visual instruction tuning and other multimodal or domain-specific adaptation settings.

In addition, the relationship between adaptation parameters and knowledge sharing across tasks or clients is not yet fully understood. In particular, it remains unclear whether there exists an optimal sparse or structured parameter-sharing pattern that balances specialization and generalization in low-rank adaptation. A deeper theoretical and empirical understanding of these mechanisms would be crucial for fully characterizing the behavior and limitations of low-rank adaptation methods. We leave these directions for future work.

Ethical Statement

All our experiments use publicly available datasets and open-source pretrained models and are intended only for research purposes. The main risks stem from dataset biases and limitations of pre-trained models in real-world applications. We do not advocate deployment without human oversight.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. 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 7319–7328.

Hao Ban and Kaiyi Ji. 2024. Fair resource allocation in multi-task learning. In *Proceedings of the 41st International Conference on Machine Learning*, pages 2715–2731.

Hao Ban, Gokul Ram Subramani, and Kaiyi Ji. 2025. SAMO: A lightweight sharpness-aware approach for multi-task optimization with joint global-local perturbation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 785–795.

Jieming Bian, Lei Wang, Letian Zhang, and Jie Xu. 2025. Fedalt: Federated fine-tuning through adaptive local training with rest-of-world lora. *arXiv preprint arXiv:2503.11880*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. 2025. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*.

Weiyu Chen, Baijiong Lin, Xiaoyuan Zhang, Xi Lin, Han Zhao, Qingfu Zhang, and James T Kwok. 2025. Gradient-based multi-objective deep learning: Algorithms, theories, applications, and beyond. *arXiv preprint arXiv:2501.10945*.

Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, and Gauri Joshi. 2024. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12903–12913.

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Dongshang Deng, Xuangou Wu, Tao Zhang, Xiangyun Tang, Hongyang Du, Jiawen Kang, Jiqiang Liu, and Dusit Niyato. 2024. Fedasa: A personalized federated learning with adaptive model aggregation for heterogeneous mobile edge computing. *IEEE Transactions on Mobile Computing*.

Jean-Antoine Désidéri. 2012. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36:10088–10115.

Enmao Diao, Jie Ding, and Vahid Tarokh. 2021. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*.

Qin Dong, Yuntian Tang, Heming Jia, Yunhang Shen, Bohan Jia, Wenxuan Huang, Lianyue Zhang, Jiao Xie, and Shaohui Lin. 2025. Masa: Rethinking the representational bottleneck in lora with multi-a shared adaptation. *arXiv preprint arXiv:2510.06005*.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, and 1 others. 2024. Loramoe: Alleviating world knowledge forgetting in large language models via moe-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, and 1 others. 2023. LoRAMoE: Alleviate world knowledge forgetting in large language models via moe-style plugin. *arXiv preprint arXiv:2312.09979*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Pengxin Guo, Shuang Zeng, Yanran Wang, Huijie Fan, Feifei Wang, and Liangqiong Qu. 2025. Selective aggregation for low-rank adaptation in federated learning. In *The Thirteenth International Conference on Learning Representations*.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Transactions on Machine Learning Research*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

Yifei He, Shiji Zhou, Guojun Zhang, Hyokun Yun, Yi Xu, Belinda Zeng, Trishul Chilimbi, and Han Zhao. 2024. Robust multi-task learning with excess risks. In *Forty-first International Conference on Machine Learning*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5276.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. Lorahub: Efficient cross-task generalization via dynamic lora composition. In *First Conference on Language Modeling*.

Minhui Huang, Dewei Zhang, and Kaiyi Ji. 2023. Achieving linear speedup in non-iid federated bilevel learning. In *International Conference on Machine Learning*, pages 14039–14059. PMLR.

Ninghui Jia, Zhihao Qu, Baoliu Ye, Yanyan Wang, Shihong Hu, and Song Guo. 2025. A comprehensive survey on communication-efficient federated learning in mobile edge environments. *IEEE Communications Surveys & Tutorials*.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491.

Samir Khaki, Xiuyu Li, Junxian Guo, Ligeng Zhu, Konstantinos N Plataniotis, Amir Yazdanbakhsh, Kurt Keutzer, Song Han, and Zhijian Liu. 2025. Sparselora: Accelerating llm fine-tuning with contextual sparsity. In *Forty-second International Conference on Machine Learning*.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157.

Tao Lei, Junwen Bai, Siddhartha Brahma, Joshua Ainslie, Kenton Lee, Yanqi Zhou, Nan Du, Vincent Zhao, Yuexin Wu, Bo Li, and 1 others. 2023. Conditional adapters: Parameter-efficient transfer learning with fast inference. *Advances in Neural Information Processing Systems*, 36:8152–8172.

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.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and 1 others. 2024. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. *arXiv preprint arXiv:2404.15159*.

Dengchun Li, Naizheng Wang, Zihao Zhang, Haoyang Yin, Lei Duan, Meng Xiao, and Mingjie Tang. 2025a. Dynmole: Boosting mixture of lora experts fine-tuning with a hybrid routing mechanism. *arXiv preprint arXiv:2504.00661*.

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.

Youqi Li, Fan Li, Song Yang, and Yu Wang. 2025b. Bgefl: Enabling communication-efficient federated learning via bandit gradient estimation in resource-constrained networks. *IEEE Transactions on Networking*.

Jian Liang, Wenke Huang, Xianda Guo, Guancheng Wan, Bo Du, and Mang Ye. 2025. Thanora: Task heterogeneity-aware multi-task low-rank adaptation. *arXiv preprint arXiv:2505.18640*.

Mengqi Liao, Wei Chen, Junfeng Shen, Shengnan Guo, and Huaiyu Wan. 2025. Hmora: Making llms more effective with hierarchical mixture of lora experts. In *The Thirteenth International Conference on Learning Representations*.

Baijiong Lin, Weisen Jiang, Pengguang Chen, Shu Liu, and Ying-Cong Chen. 2025. Mtmamba++: Enhancing multi-task dense scene understanding via mamba-based decoders. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor Tsang. 2022. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*.

Baijiong Lin and Yu Zhang. 2023. Libmtl: A python library for deep multi-task learning. *Journal of Machine Learning Research*, 24(209):1–7.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. 2021. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*.

Xuezheng Liu, Yipeng Zhou, Di Wu, Miao Hu, Min Chen, Mohsen Guizani, and Quan Z Sheng. 2025. Cpfedavg: Enhancing hierarchical federated learning via optimized local aggregation and parameter mixing. *IEEE Transactions on Networking*.

Guodong Long, Tao Shen, Jing Jiang, Michael Blumenstein, and 1 others. 2024. Dual-personalizing adapter for federated foundation models. *Advances in Neural Information Processing Systems*, 37:39409–39433.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2402.12851*.

Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. 2019. Attentive single-tasking of multiple tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1851–1860.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR.

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

Mahdi Morafah, Saeed Vahidian, Weijia Wang, and Bill Lin. 2023. Flis: Clustered federated learning via inference similarity for non-iid data distribution. *IEEE Open Journal of the Computer Society*, 4:109–120.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, and 1 others. 2023. Crosslingual generalization through multitask finetuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15991–16111.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. 2022. Multi-task learning as a bargaining game. In *International Conference on Machine Learning*, pages 16428–16446. PMLR.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094.

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.

Hoang Phan, Lam Tran, Quyen Tran, Ngoc Tran, Tuan Truong, Qi Lei, Nhat Ho, Dinh Phung, and Trung Le. 2025. Beyond losses reweighting: Empowering multi-task learning via the generalization perspective. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2440–2450.

Arian Raje, Baris Askin, Divyansh Jhunjhunwala, and Gauri Joshi. 2025. Ravan: Multi-head low-rank adaptation for federated fine-tuning. *arXiv preprint arXiv:2506.05568*.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4822–4829.

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. 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 4463–4473.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *Advances in Neural Information Processing Systems*, 31.

Mengyang Sun, Yihao Wang, Tao Feng, Dan Zhang, Yifan Zhu, and Jie Tang. 2025. A stronger mixture of low-rank experts for fine-tuning foundation models. In *Forty-second International Conference on Machine Learning*.

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.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Cheng-Zhong Xu. 2024. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37:9565–9584.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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.

Qifan Wang, Yuning Mao, Jingang Wang, Hanchao Yu, Shaoliang Nie, Sinong Wang, Fuli Feng, Lifu Huang, Xiaojun Quan, Zenglin Xu, and 1 others. 2023. Aprompt: Attention prompt tuning for efficient adaptation of pre-trained language models.

In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9147–9160.

Zheng Wang, Zihui Wang, Xiaoliang Fan, and Cheng Wang. 2025. Federated learning with domain shift eraser. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 4978–4987.

Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. 2024. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations. *Advances in Neural Information Processing Systems*, 37:22513–22533.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. 2024a. Mixture-of-subspaces in low-rank adaptation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7880–7899.

Xun Wu, Shaohan Huang, and Furu Wei. 2024b. Mixture of lora experts. In *The Twelfth International Conference on Learning Representations*.

Peiyao Xiao, Hao Ban, and Kaiyi Ji. 2023. Direction-oriented multi-objective learning: Simple and provable stochastic algorithms. *Advances in Neural Information Processing Systems*, 36:4509–4533.

Peiyao Xiao, Chaosheng Dong, Shaofeng Zou, and Kaiyi Ji. 2025. Ldc-mtl: Balancing multi-task learning through scalable loss discrepancy control. *arXiv preprint arXiv:2502.08585*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Menglin Yang, Jialin Chen, Yifei Zhang, Jiahong Liu, Jiasheng Zhang, Qiyao Ma, Harshit Verma, Qianru Zhang, Min Zhou, Irwin King, and 1 others. 2024. Low-rank adaptation for foundation models: A comprehensive review. *arXiv preprint arXiv:2501.00365*.

Yaming Yang, Dilxat Muhtar, Yelong Shen, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Weiwei Deng, Feng Sun, Qi Zhang, and 1 others. 2025b. Mtl-lora: Low-rank adaptation for multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 22010–22018.

Shih-Ying Yeh, Yu-Guan Hsieh, Zhidong Gao, Bernard BW Yang, Giyeong Oh, and Yanmin Gong. 2023. Navigating text-to-image customization: From lycoris fine-tuning to model evaluation. In *The Twelfth International Conference on Learning Representations*.

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

Dacao Zhang, Kun Zhang, Shimao Chu, Le Wu, Xin Li, and Si Wei. 2025a. More: A mixture of low-rank experts for adaptive multi-task learning. *arXiv preprint arXiv:2505.22694*.

Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Tong Yu, Guoyin Wang, and Yiran Chen. 2024. Towards building the federatedgpt: Federated instruction tuning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6915–6919. IEEE.

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

Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609.

Zhen-Ru Zhang, Chuanqi Tan, Haiyang Xu, Chengyu Wang, Jun Huang, and Songfang Huang. 2023b. Towards adaptive prefix tuning for parameter-efficient language model fine-tuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1239–1248.

Zikai Zhang, Ping Liu, Jiahao Xu, and Rui Hu. 2025b. Fed-hello: Efficient federated foundation model fine-tuning with heterogeneous lora allocation. *arXiv preprint arXiv:2506.12213*.

Ziyu Zhao, Yixiao Zhou, Zhi Zhang, Didi Zhu, Tao Shen, Zexi Li, Jinluan Yang, Xuwu Wang, Jing Su, Kun Kuang, and 1 others. 2025. Each rank could be an expert: Single-ranked mixture of experts lora for multi-task learning. *arXiv preprint arXiv:2501.15103*.

Yiyun Zhou, Chang Yao, and Jingyuan Chen. 2025. CoLA: Collaborative low-rank adaptation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14115–14130.

Peizhen Zhu and Andrew V Knyazev. 2013. Angles between subspaces and their tangents. *Journal of Numerical Mathematics*, 21(4):325–340.

Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. 2023. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices. *arXiv preprint arXiv:2309.02411*.

A Additional Related Work

Parameter-efficient fine-tuning. PEFT adapts large language models (LLM) by training only a small subset of parameters, significantly reducing the computation cost while maintaining the performance compared to full fine-tuning. Existing PEFT methods are typically categorized into the following groups. (1) Adapter-based methods, which insert small trainable modules either serially or in parallel with Transformer blocks while keeping the pre-trained model frozen (Houlsby et al., 2019; He et al., 2022; Lei et al., 2023; Pfeiffer et al., 2021). (2) Soft prompt-based methods, such as prefix tuning (Li and Liang, 2021; Zhang et al., 2023b) and prompt tuning (Lester et al., 2021; Wang et al., 2023), which prepend learnable embeddings to the input without modifying model weights. (3) Low rank adaptation (LoRA) methods (Hu et al., 2022; Dettmers et al., 2023; Liu et al., 2024), which approximately reparameterize weight updates using low-rank matrices. These updates can be merged into frozen weights, thereby introducing no additional inference latency.

Multi-task learning. The primary goal of MTL is to promote positive knowledge transfer while reducing negative interference across tasks (Zhang and Yang, 2021; Lin and Zhang, 2023; Chen et al., 2025). Existing methods are commonly divided into three categories. (1) Soft-parameter sharing, which encourages task parameters to be similar but not identical, allowing flexible knowledge transfer while maintaining distinctions. Ruder et al. (2019) propose learning a latent meta-architecture for multiple tasks, where task layers are regularized by their distance to the latent model. (2) Modularity-based methods, which employ task-specific modules to mitigate negative transfer. For example, Lin et al. (2025) integrate task-specific modules to avoid gradient conflicts. (3) Task-balancing optimization methods adjust task weights or gradients using loss or gradient information. These methods include task loss weighting (Kendall et al., 2018; Lin et al., 2022; Xiao et al., 2025) and gradient weighting approaches, such as MGDA (Désidéri, 2012; Sener and Koltun, 2018) and its variants (Liu et al., 2021; Xiao et al., 2023), which apply multi-objective optimization to explore Pareto-optimal solutions for balancing gradient conflicts. Other methods examine MTL from the perspective of label noise (He et al., 2024), fairness (Navon et al., 2022; Ban and Ji, 2024) or analyze the sharpness of the loss landscape (Phan et al., 2025; Ban et al., 2025).

Federated learning. FL allows multiple clients to train collaboratively in a decentralized setting while preserving data privacy by keeping raw data local. In each communication round, clients compute local updates on their private data and send only these updates (e.g., weight deltas or gradients) to a server, which aggregates them into a new global model. Existing methods are typically grouped into three categories. (1) Aggregation methods, which improve how client updates are combined, such as robust aggregation or matched averaging (Morafah et al., 2023; Huang et al., 2023; Liu et al., 2025). (2) Communication-efficient methods, which reduce the cost of transmitting updates through techniques like sparsification, or partial updates (Jia et al., 2025; Li et al., 2025b). (3) Personalization methods, which adapt the global model to each client's unique data distribution, using approaches such as model splitting, or client-specific parameters (Diao et al., 2021; Deng et al., 2024; Wang et al., 2025).

B Additional Details for Section 3

B.1 Similarity Metric in LoRA Modules

LoRA represents the weight updates as $\Delta W = BA$ with $A \in \mathbb{R}^{r\times d_{\text{in}}}$, $B \in \mathbb{R}^{d_{\text{out}}\times r}$. For any invertible $R \in \mathbb{R}^{r\times r}$, we have

$$\Delta W = BA = (BR)(R^{-1}A).$$

This shows that A and B are not individually unique. They can be arbitrarily rotated within the rank-r subspace without changing ΔW. As a result, directly computing the cosine similarity between A or B matrices can give misleading results.

Different seeds or initializations may lead to very different A and B, but the subspaces they span are rotation-invariant. If the subspaces align, the modules are functionally aligned. Therefore, we use the subspace similarity proposed by Zhu and Knyazev (2013). Specifically, given two matrices $M_1, M_2 \in \mathbb{R}^{d\times r}$, we compute SVD of each and obtain the orthonormal bases of their column spaces, $U_1, U_2 \in \mathbb{R}^{r\times r}$. The similarity is then defined as

$$\text{Sim}(M_1, M_2) = \frac{1}{r}\|U_1^\top U_2\|_F^2 \in [0, 1],$$

where a higher value indicates a stronger alignment.

B.2 Analysis in Dynamics of LoRA Modules

Section 3.1-3.2 analyze the learning behavior of A and B matrices during fine-tuning on different tasks based on LLaMA2-7B. We also perform the same analysis in the federated fine-tuning with two clients. Following Zhang et al. (2024), we randomly sample data from the Dolly-15K dataset (Conover et al., 2023) and split them into two clients, each containing 1493 instruction data samples from different tasks. The data distribution is shown in Figure 8.

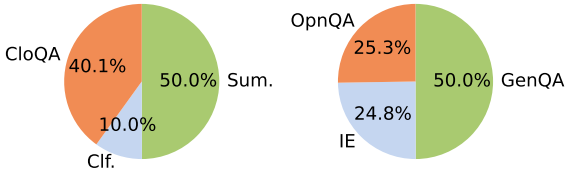
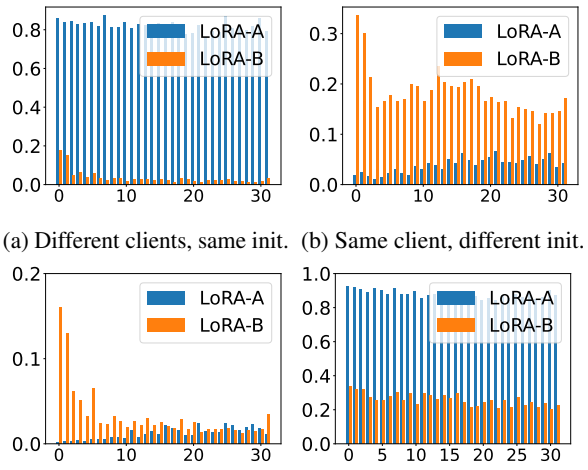


Figure 8: Data distribution of clients. Left: Client 1 contains Closed QA, Summarization, and Classification tasks. Right: Client 2 contains Open QA, General QA, and Information Extraction tasks.

We fine-tune the LLaMA2-7B model on the two clients and analyze how the similarity of LoRA modules is affected by random seeds for initialization. Using the similarity metric described in Appendix B.1, we compute the layer-wise similarity of LoRA modules across the two clients. The results, shown in Figure 9a-9c, indicate that, contrary to the assumption in FedSA-LoRA (Guo et al., 2025), the similarity of A matrices across clients comes mainly from identical initialization rather than shared knowledge.

Furthermore, we analyze the learning dynamics of A and B in the federated fine-tuning setting. On client 2, we compute the similarity of A before and after fine-tuning, and do the same for B. We also calculate the magnitude and direction variation for this client. The results, shown in Figure 9d-10, are consistent with our earlier fine-tuning experiments on different tasks. They confirm that B plays a more critical role than A in encoding knowledge across clients.

To further explore whether the above observation depends on the model or dataset, we analyze the learning dynamics of LoRA the Qwen2-7B model using the bigscience/xP3 dataset (Muennighoff et al., 2023), which contains data from 46 languages and 16 NLP tasks. We sample 3,000 English examples and fine-tune the model using the same configuration. The results are shown in Figure 11. We observe the same pattern: the B matrix plays a more dominant role than A matrix during training.



(a) Different clients, same init. (b) Same client, different init.

(c) Different clients and init. (d) Before and after FT.

Figure 9: Layer-wise similarity analysis. The x-axis denotes the layer indices, and the y-axis denotes the similarity scores. Subfigures (a)-(c) compare different LoRA modules under clients and initialization: (a) two different clients with the same initialization; (b) the same client with different initializations; and(c) two different clients with different initializations. Subfigure (d) compares the same LoRA module before and after fine-tuning. A matrices are similar only under the same initialization, whereas B exhibits relatively stable similarity across different tasks and seeds. In addition, A remains largely unchanged from initialization.

B.3 Practical Implementation

For the results in Section 3.1–3.2, we fine-tune LLaMA2-7B and Qwen2-7B for 3 epochs. The training uses a learning rate of 3e-4, batch size 32, and gradient accumulation step 2. We follow the alpaca_short template (Taori et al., 2023) to construct the instruction data. LoRA is applied to the q_{proj} modules with rank $r = 8$.

For the analysis of lazy learning in multi-task fine-tuning (Section 3.3), we fine-tune the LLaMA3-8B model on the commonsense reasoning 15K dataset (Hu et al., 2023) for 3 epochs, using a learning rate of 3e-4, batch size 4, and gradient accumulation step 4. LoRA is applied to the q_{proj} modules with rank $r = 8$. The sharing-A structure uses 3 A matrices, and the sharing-B structure uses 3 B matrices.

For the analysis of knowledge transfer in federated fine-tuning (Section 3.3), we fine-tune the LLaMA2-7B model on the federated NLP dataset (Long et al., 2024) for 10 communication rounds, with 10 local epochs per client. The learning rate

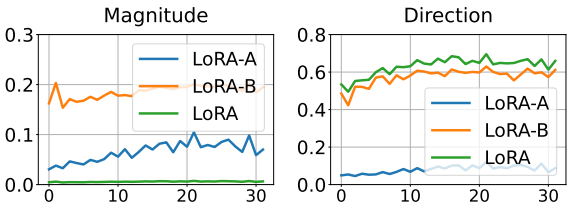


Figure 10: Comparison of LoRA modules on each client before and after federated fine-tuning. The x-axis denotes the layer indices, and the y-axis denotes the variation values. Left: magnitude change; Right: direction change. LoRA shows a limited magnitude change, with nearly all directional change captured by B.

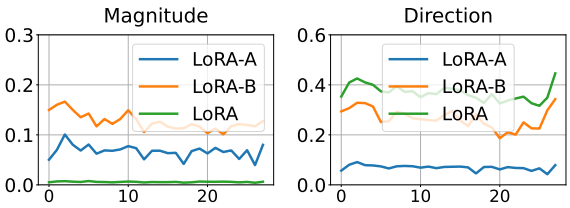


Figure 11: Comparison of LoRA modules using Qwen2-7B before and after fine-tuning. Left: magnitude change; Right: direction change.

is 5e-4, the batch size is 32, and the gradient accumulation step is 2. In the homogeneous setting, all clients use rank 8. In the heterogeneous setting, the ranks are set to $\{64, 64, 32, 32, 16, 16, 8, 8\}$. The methods are applied to q_{proj} and v_{proj} modules. All experiments are conducted on RTX A6000 GPU.

C Additional Experimental Details and Results

C.1 Benchmarks

The intra-domain multi-task commonsense reasoning 170K benchmark contains questions from the following datasets: (1) ARC-Challenge and ARC-Easy (Clark et al., 2018), which consist of grade-school–level multiple-choice science questions; (2) BoolQ (Clark et al., 2019), a yes/no question-answering dataset requiring non-factoid reasoning and entailment; (3) HellaSwag (Zellers et al., 2019), a dataset of commonsense natural language inference (NLI) questions that require identifying the most appropriate continuation of a narrative input; (4) OpenBookQA (Mihaylov et al., 2018), which contains questions requiring multi-step reasoning by combining provided scientific facts with external background knowledge; (5) PIQA (Bisk et al., 2020), a dataset of everyday commonsense reasoning questions about the physical world; (6) SIQA (Sap et al., 2019), which focuses on social and emotional commonsense reasoning in everyday human interactions; (7) WinoGrande (Sakaguchi et al., 2021), a collection of fill-in-the-blank sentences designed to test pronoun resolution using commonsense.

The cross-domain multi-task NLP dataset contains 8 NLP tasks sampled from the FLAN dataset (Wei et al., 2022). The tasks are: (1) Commonsense, a reasoning task that requires everyday knowledge to make judgments; (2) Entailment, an NLI task that determines the relationship between a premise and a hypothesis; (3) Open-domain QA, a question answering task that retrieves or generates answers from open sources; (4) Paraphrase, a classification task that recognizes whether a sentence pair is semantically equivalent; (5) Reading comprehension, a question answering task requires understanding the text content and answering the related questions; (6) Sentiment classification, a classification task that determines the whether the sentiment polarity is neutral, positive, or negative; (7) Summarization, an NLG task that produces a compact digest of a long passage while keeping the critical information; (8) Text formatting, an NLG task that corrects the punctuation in unformatted text. Each task has 300 examples for training and 200 examples for testing.

The federated NLP dataset also contains 8 NLP tasks sampled from the FLAN dataset (Wei et al., 2022). The tasks are: (1) Coreference, a discourse understanding task that requires determining which entity a pronoun refers to; (2) Entailment, an NLI task that determines the relationship between a premise and a hypothesis; (3) Linguistic Acceptability, a classification task that detects whether a sentence is grammatical; (4) Paraphrase, a classification task that recognizes whether a sentence pair is semantically equivalent; (5) Question classification, a task for question understanding in question answering systems; (6) Structure-to-Text, a natural language generation (NLG) task that converts structured triples into natural language; (7) Text formatting, an NLG task that corrects the punctuation in unformatted text; (8) Word sense disambiguation, a classification task that determines whether the same word has the same meaning in two different sentences. Each task has 300 examples for training and 200 examples for testing, and we assign one task to each client.

For the intra-domain multi-task fine-tuning, we also consider the math reasoning 10K benchmark (Hu et al., 2023), which includes 4 datasets: (1) AQuA (Ling et al., 2017), which contains multiple-choice algebra word problems, each accompanied

by a natural language rationale explaining the step-by-step reasoning; (2) GSM8K (Cobbe et al., 2021), a high-quality collection of linguistically diverse grade-school–level math word problems designed to evaluate multi-step reasoning; (3) MAWPS (Koncel-Kedziorski et al., 2016), a compilation of math word problems intended to support robust and scalable research on arithmetic reasoning, including AddSub (basic addition/subtraction), SingleOp (single-operator arithmetic), MultiArith (multi-step arithmetic), SingleEq (single-equation algebra); (4) SVAMP (Patel et al., 2021), which consists of simple one-unknown grade-school–level arithmetic word problems, designed to test robustness in arithmetic reasoning.

The original split of the math reasoning 10K benchmark is not suitable for multi-task fine-tuning, since it does not include the full training data of the subsidiary tasks. In addition, Hu et al. (2023) report data leakage issues in this benchmark. To address these concerns, we downloaded the original data of each single task, and checked every training example in the benchmark to determine whether it belongs to the training set of any individual task. This process allowed us to construct a training dataset for each task, making it possible to fine-tune on single tasks and obtain single-task baselines. For multi-task fine-tuning, we fine-tune directly on the benchmark and evaluate on each task. Since the single-task training splits are created by us, we report the corresponding results in the Appendix.

C.2 Baselines

For multi-task fine-tuning, we compare our ALoRA with the following methods: (1) the vanilla LoRA (Hu et al., 2022); (2) LoHa (Yeh et al., 2023), which employs Hadamard decompositions to the updates; (3) AdaLoRA (Zhang et al., 2023a), which adaptively prunes rank using SVD; (4) MoSLoRA (Wu et al., 2024a), which introduces an additional matrix to fuse update subspaces; (5) HydraLoRA (Tian et al., 2024), which adopts a sharing-A multi-LoRA framework.

For homogeneous federated fine-tuning, we compare our Fed-ALoRA with the following methods: (1) FedIT (Zhang et al., 2024), where each client transmits the full LoRA parameters; (2) FedDPA (Long et al., 2024), which employs both a global adapter and a local adapter for each client; (3) FedSA-LoRA (Guo et al., 2025), which shares only the A matrices. For the heterogeneous federated fine-tuning, we compare Fed-ALoRA with: (1) ZeroPadding, which pads all heterogeneous ranks to the maximum rank across clients, enabling FedIT to support heterogeneity; (2) FLoRA (Wang et al., 2024), a stacking-based noise-free aggregation method; (3) FedSA-LoRA (Guo et al., 2025), which does not natively support heterogeneity but is adapted here using the decomposition proposed in our method.

C.3 Practical Implementation

For intra-domain multi-task fine-tuning on commonsense reasoning and math reasoning, we fine-tune LLaMA3-8B and Qwen2-7B models for 3 epochs on the training data with a learning rate of 3e-4. The AdamW optimizer is used with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The batch size is 4, and the gradient accumulation step is 4. For HydraLoRA, we follow the original setup with one single A matrix and 3 B matrices with rank 8. For our ALoRA, we use 3 A matrices and a single B matrix with rank 8. To ensure a fair comparison, the other baselines are configured with a comparable parameter size: LoRA, LoHa, and MoSLoRA use rank 16, while others use rank 8. All methods are applied to the q_{proj} and o_{proj} modules.

For cross-domain multi-task fine-tuning, we fine-tune LLaMA2-7B and Qwen2-7B models for 50 epochs on the training data with a learning rate of 3e-4. The AdamW optimizer is used with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The batch size is 4, and the gradient accumulation step is 4. For HydraLoRA, we follow the original setup with one single A matrix and 3 B matrices with rank 8. For our ALoRA, we use 3 A matrices and a single B matrix with rank 8. To ensure a fair comparison, the other baselines are configured with a comparable parameter size: LoRA, LoHa, and MoSLoRA use rank 16, while others use rank 8. All methods are applied to the $(q_{\text{proj}}, v_{\text{proj}})$ modules.

For federated fine-tuning, we fine-tune LLaMA2-7B and Qwen2-7B models for 10 communication rounds with 10 local epochs per client, using a learning rate of 5e-4. The AdamW optimizer is used with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The batch size is 32, and the gradient accumulation step is 2. In the homogeneous setting, all clients use rank 8. In the heterogeneous setting, the ranks are set to $\{64, 64, 32, 32, 16, 16, 8, 8\}$, and d_m is chosen to 16. All methods are applied to $(q_{\text{proj}}, v_{\text{proj}})$ modules for LLaMA2-7B and to $(q_{\text{proj}}, o_{\text{proj}})$ modules for Qwen2-7B model. All experiments are conducted

on RTX A6000 GPU.

C.4 Full Results on Sharing-A and Sharing-B

To assess which parameters should be shared to improve cross-client knowledge transder, we compare the performance of each client on all tasks between FedSA and our Fed-ALoRA in Section 3.3. The full results are presented in Table 7.

C.5 Full Results on Commonsense Reasoning

We provide the full experiment results on the commonsense reasoning benchmark, including the per-task performance, mean and standard deviations of 3 independent runs. The details are shown in Table 8.

C.6 Full Results on Multi-Task NLP Datasets

We provide the full experiment results on the cross-domain multi-task NLP datasets, including the per-task performance. The details are shown in Table 9.

C.7 Full Results on Homogeneous Federated Learning

We provide the full experiment results on the homogeneous federated learning, including the per-client performance. The details are shown in Table 10.

C.8 Full Results on Heterogeneous Federated Learning

We provide the full experiment results on the heterogeneous federated learning, including per-client performance. We implement the Sharing-A method based on our decomposition strategy for dealing with varying clients. The details are shown in Table 11.

C.9 Full Results on the Intermediate Rank

We also provide the full results of the study of different intermediate ranks using LLaMA2-7B model in Section 5.3, which are shown in Table 12.

C.10 Comparison with Additional Multi-Task Fine-Tuning Methods

We compare ALoRA with existing multi-task fine-tuning methods that explicitly utilizing task-specific information. Since MASA has not publicly released its source code, we are unable to include it in our comparison. The results shown in Table 13 further demonstrate the effectiveness of our method.

C.11 Comparison Between HydraLoRA and ALoRA at Larger Model Scales

To further validate the benefits of sharing B at a larger model scale, we conduct experiments on commonsense benchmark using Qwen2.5-14B. We adopt a learning rate of 3e-5 and apply adaptation to the q_{proj} and o_{proj} modules. The results are shown in Table 14. They show that ALoRA consistently outperforms HydraLoRA.

C.12 Additional Results on Math Reasoning

The results of LLaMA2-7B model on the math reasoning benchmark are presented in Table 15. LoRA, MoSLoRA, and our ALoRA outperform the single-task baseline on all tasks, but ALoRA achieves the most balanced performance, showing that it enables more effective knowledge transfer than the baselines.

D The Use of Large Language Models

In preparing this manuscript, large language models (LLMs) were used only to assist with language polishing and stylistic refinement. All technical content, formulations, experimental designs, and conceptual contributions were developed by the authors. Importantly, LLMs were not used for ideation and methodology development.

Setting	Method	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8	Avg.
Homogeneous	Sharing A	50.02	49.78	59.92	42.66	54.05	25.21	23.32	49.43	44.30
	Sharing B	67.43	67.83	69.85	69.26	69.50	60.94	54.86	70.90	66.32
Heterogeneous	Sharing A	43.01	41.58	50.09	38.34	53.53	24.82	24.28	50.41	40.76
	Sharing B	48.38	54.80	58.30	46.98	62.15	35.09	31.80	64.89	50.30

Table 7: Comparing sharing A versus B in federated fine-tuning. Sharing B consistently outperforms sharing A in both the homogeneous and heterogeneous settings.

Method	ARC-C	ARC-E	BoolQ	HellaS.	OBQA	PIQA	SIQA	WinoG.	Avg.	$\Delta m\%(\downarrow)$
LLaMA3-8B										
ST Baseline	77.76 $\pm$0.97	90.81 $\pm$0.23	73.39 $\pm$1.05	95.40 $\pm$0.08	86.60 $\pm$0.72	89.25 $\pm$0.47	80.69 $\pm$0.48	85.35 $\pm$0.64	84.90 $\pm$0.21	
LoRA	76.66 $\pm$1.02	88.72 $\pm$0.83	72.73 $\pm$1.24	94.50 $\pm$0.41	84.10 $\pm$0.14	87.59 $\pm$0.15	79.42 $\pm$0.15	85.40 $\pm$1.00	83.64 $\pm$0.27	1.48
LoHa	77.13 $\pm$0.13	88.91 $\pm$0.09	72.89 $\pm$0.19	94.05 $\pm$0.23	85.40 $\pm$0.56	87.84 $\pm$0.34	78.92 $\pm$0.44	84.73 $\pm$0.17	83.73 $\pm$0.11	1.36
AdaLoRA	77.72 $\pm$0.72	89.72 $\pm$0.96	73.32 $\pm$0.19	93.98 $\pm$0.73	85.20 $\pm$1.13	87.90 $\pm$0.35	79.45 $\pm$0.18	83.78 $\pm$0.28	83.88 $\pm$0.32	1.17
MoSLoRA	76.87 $\pm$0.12	89.22 $\pm$0.15	73.50 $\pm$0.86	95.02 $\pm$0.13	85.27 $\pm$2.58	87.99 $\pm$0.59	80.89 $\pm$0.12	85.13 $\pm$0.55	84.23 $\pm$0.39	0.76
HydraLoRA	78.58 $\pm$0.12	89.94 $\pm$0.18	75.02 $\pm$0.18	95.21 $\pm$0.11	84.90 $\pm$1.55	88.03 $\pm$0.15	79.99 $\pm$0.72	84.92 $\pm$0.34	84.57 $\pm$0.17	0.32
ALoRA (ours)	79.40 $\pm$0.41	89.69 $\pm$0.71	74.38 $\pm$0.10	94.85 $\pm$0.23	86.10 $\pm$0.14	88.24 $\pm$0.49	80.17 $\pm$0.32	85.68 $\pm$0.27	**84.81** $\pm$0.03	**0.04**
Qwen2-7B										
ST Baseline	86.40 $\pm$0.38	94.01 $\pm$0.14	74.04 $\pm$1.01	95.54 $\pm$0.25	91.33 $\pm$0.83	90.91 $\pm$0.33	81.94 $\pm$0.15	83.97 $\pm$0.85	87.27 $\pm$0.24	
LoRA	83.30 $\pm$0.55	92.95 $\pm$0.12	74.37 $\pm$0.40	93.68 $\pm$1.20	88.46 $\pm$1.22	89.13 $\pm$0.49	80.48 $\pm$0.29	85.34 $\pm$0.43	85.96 $\pm$0.31	1.43
LoHa	85.22 $\pm$0.72	94.03 $\pm$0.28	73.58 $\pm$0.11	94.38 $\pm$0.52	88.53 $\pm$0.61	89.85 $\pm$0.37	80.31 $\pm$0.22	84.76 $\pm$0.85	86.34 $\pm$0.33	1.06
AdaLoRA	85.39 $\pm$0.05	93.86 $\pm$0.06	72.61 $\pm$0.35	94.09 $\pm$0.30	89.60 $\pm$0.20	89.13 $\pm$0.03	80.95 $\pm$0.11	83.50 $\pm$0.83	86.14 $\pm$0.17	1.30
MoSLoRA	84.80 $\pm$0.64	93.18 $\pm$0.19	73.33 $\pm$1.13	94.02 $\pm$0.71	89.80 $\pm$0.53	89.66 $\pm$0.34	80.58 $\pm$0.67	85.05 $\pm$0.81	86.30 $\pm$0.16	1.09
HydraLoRA	84.60 $\pm$0.30	93.20 $\pm$0.03	73.37 $\pm$1.42	94.42 $\pm$0.37	88.30 $\pm$0.42	89.58 $\pm$0.73	80.97 $\pm$0.06	84.33 $\pm$0.95	86.09 $\pm$0.42	1.32
ALoRA (ours)	85.28 $\pm$0.42	93.69 $\pm$0.29	73.41 $\pm$0.15	94.76 $\pm$0.36	90.10 $\pm$0.42	89.07 $\pm$0.07	80.94 $\pm$0.47	84.50 $\pm$0.50	**86.47** $\pm$0.07	**0.91**

Table 8: Results on intra-domain multi-task commonsense reasoning benchmark. $\Delta m\%$ measures performance balance across tasks. $\downarrow$ denotes that lower values are better. All methods use the same number of adapter parameters. We run each experiment 3 times and report the average with the error bar.

Method	CSR	Ent	ODQA	Para	RC	Sent	Sum	TFmt	Avg.	$\Delta m\%$
LLaMA2-7B										
ST Baseline	45.15	65.00	75.19	55.00	78.00	71.75	28.17	88.6	63.36	
LoRA	53.19	63.00	84.31	51.00	50.50	69.50	32.53	89.36	61.67	0.31
LoHa	49.94	60.94	79.78	67.70	69.57	73.50	33.33	90.85	65.70	-5.76
AdaLoRA	51.94	56.94	78.57	65.22	66.61	59.50	31.95	84.93	61.96	-0.41
MoSLoRA	50.70	60.50	81.11	71.50	70.00	75.00	32.64	87.95	66.18	-6.58
HydraLoRA	44.51	67.50	75.83	74.50	76.50	71.50	32.14	89.10	66.45	-6.39
ALoRA	48.21	62.50	80.35	78.50	68.50	75.00	33.79	90.20	**67.13**	**-8.33**
Qwen2-7B										
ST Baseline	76.30	90.00	90.95	87.5	77.50	71.00	25.46	90.59	76.16	
LoRA	87.20	86.00	94.41	86.00	80.50	76.50	32.35	93.87	79.60	-6.78
LoHa	84.52	89.00	93.65	82.00	76.50	76.00	32.54	92.62	78.35	-5.27
AdaLoRA	81.11	87.00	93.92	82.50	74.55	76.00	32.57	93.26	77.61	-4.33
MoSLoRA	84.75	89.50	94.63	86.00	80.06	72.50	33.26	94.02	79.34	-6.59
HydraLoRA	88.16	87.00	93.84	88.50	80.23	76.50	31.95	94.04	80.03	-7.14
ALoRA	84.99	89.50	94.79	85.50	81.50	80.00	32.77	94.65	**80.46**	**-7.98**

Table 9: Results on cross-domain multi-task NLP datasets. $\Delta m\%$ measures performance.

Method	Coref	Ent	LAcc	Para	QCls	S2T	TFmt	WSD	Avg.	$\Delta m\%$	Params.
LLaMA2-7B											
ST Baseline	73.00	84.00	79.00	78.00	94.00	72.21	96.64	60.50	79.67		
FedIT	86.24	86.50	78.00	81.00	94.50	72.06	96.51	65.00	82.47	-3.92	8.39
FedDPA	88.51	85.50	73.50	77.50	95.50	73.76	96.40	65.00	81.96	-3.30	16.78
FedSA-LoRA	81.77	86.00	78.00	75.00	93.50	73.34	96.55	65.00	81.15	-2.21	**4.19**
Fed-ALoRA	85.74	87.00	73.50	79.00	94.00	73.10	96.24	71.50	**82.51**	**-4.29**	**4.19**
Qwen2-7B											
ST Baseline	84.08	91.50	80.00	78.00	91.00	72.83	97.11	67.50	82.75		
FedIT	82.96	92.00	79.00	82.00	91.00	71.60	96.86	67.00	82.80	-0.05	6.42
FedDPA	83.41	93.50	80.00	79.50	88.00	73.83	97.58	69.50	83.17	-0.60	12.85
FedSA-LoRA	85.58	91.00	82.50	79.50	93.00	72.76	97.18	68.00	83.69	-1.15	**3.21**
Fed-ALoRA	84.02	91.50	80.00	83.50	93.00	73.64	97.27	71.50	**84.30**	**-2.05**	**3.21**

Table 10: Results for the **homogeneous** federated setting. Params. denotes the average number of parameters (in millions) transmitted per client in each round. ALoRA achieves the most balanced performance while reducing communication cost by 50% compared to full LoRA aggregation FedIT.

Method	Coref	Ent	LAcc	Para	QCls	S2T	TFmt	WSD	Avg.	$\Delta m\%$	Params.
LLaMA2-7B											
ST Baseline	81.62	88.00	81.00	79.50	94.50	72.07	96.64	60.50	81.73		
ZeroPadding	86.95	87.00	77.50	79.50	94.00	72.87	96.46	64.00	82.29	-0.91	49.28
FLoRA	82.03	87.50	75.50	74.00	95.50	70.99	96.07	62.00	80.45	1.54	141.56
Sharing-A	80.26	83.50	76.50	76.00	93.00	73.49	96.61	54.00	79.17	3.39	**12.12**
Fed-ALoRA	88.27	89.50	79.50	77.00	94.00	72.35	96.37	63.00	**82.50**	**-1.07**	**12.12**
Qwen2-7B											
ST Baseline	84.25	90.50	81.00	82.00	92.00	73.05	97.11	67.50	83.43		
ZeroPadding	84.07	92.50	80.50	79.01	90.00	73.01	97.38	70.05	83.32	0.06	37.73
FLoRA	79.98	92.00	80.00	77.00	88.00	71.04	97.19	68.00	81.65	2.13	108.38
Sharing-A	85.16	90.50	79.50	78.00	90.50	74.32	97.43	69.00	83.05	0.37	**9.23**
Fed-ALoRA	85.50	91.50	82.50	80.02	90.00	74.49	97.51	71.50	**84.13**	**-1.02**	**9.23**

Table 11: Results for the **heterogeneous** setting. ALoRA achieves the most balanced performance while reducing communication cost by 75% compared to full LoRA aggregation ZeroPadding. The original FedSA-LoRA does not support heterogeneity; Sharing-A denotes implementation with our decomposition strategy.

Fed-ALoRA	Coref	Ent	LAcc	Para	QCls	S2T	TFmt	WSD	Avg.	$\Delta m\%(\downarrow)$
$d_m = 8$	90.75	84.50	79.50	73.50	95.00	73.05	96.09	63.00	81.92	-0.41
$d_m = 16$	88.27	89.50	79.50	77.00	94.00	72.35	96.37	63.00	**82.50**	**-1.07**
$d_m = 32$	88.63	86.50	78.00	79.00	94.00	72.27	96.60	63.00	82.25	-0.80
$d_m = 64$	85.70	89.00	81.00	75.50	94.00	72.31	96.40	65.00	82.37	-1.01

Table 12: Results of LLaMA2-7B on different intermediate ranks in the heterogeneous setting.

Method	ARC-C	ARC-E	BoolQ	HellaS.	OBQA	PIQA	SIQA	WinoG.	Avg.	$\Delta m\%(\downarrow)$
ST Baseline	77.76	90.81	73.39	95.40	86.60	89.25	80.69	85.35	84.90	
LoRAMoE	76.53	88.01	73.34	93.97	83.80	86.24	80.31	83.57	83.22	1.92
MTL-LoRA	79.69	89.50	74.47	94.88	85.81	87.95	80.67	85.72	84.84	-0.001
CoLA	79.30	89.09	74.17	94.78	85.19	87.48	79.61	85.06	84.34	0.60
ALoRA (ours)	79.40	89.69	74.38	94.85	86.10	88.24	80.17	85.68	84.81	0.04

Table 13: Comparison on commonsense benchmarks between ALoRA and multi-task fine-tuning methods that explicitly use task information. Experiments are conducted on LLaMA3-8B.

Method	ARC-C	ARC-E	BoolQ	HellaS.	OBQA	PIQA	SIQA	WinoG.	Avg.
HydraLoRA	92.19	97.43	74.89	95.50	91.41	92.76	82.62	89.24	89.50
ALoRA (ours)	92.24	97.58	74.86	95.74	91.60	92.82	82.75	89.15	89.59

Table 14: Comparison between ALoRA and HydraLoRA on commonsense benchmarks at a larger model scale.

Method	AQuA	GSM8K	SVAMP	MAWPS	SingleEq	Avg.	$\Delta m\%(\downarrow)$
ST Baseline	28.34	63.68	71.10	86.13	90.75	68.00	
LoRA	30.31	66.26	75.10	90.34	94.88	71.38	-5.21
LoHa	27.56	63.84	76.70	90.76	94.88	70.75	-3.06
AdaLoRA	25.20	59.44	72.60	86.55	91.93	67.14	2.77
MoSLoRA	28.74	67.40	77.10	89.08	94.06	71.28	-4.54
HydraLoRA	27.17	68.76	75.60	90.34	93.31	71.04	-3.58
ALoRA	29.53	67.17	77.40	89.50	94.49	**71.62**	**-5.31**

Table 15: Results of LLaMA2-7B on math reasoning. $\Delta m\%$ measures performance balance across tasks. ALoRA achieves the most balanced results.