MALoRA: Mixture of Asymmetric Low-Rank Adaptation for Enhanced Multi-Task Learning

Xujia Wang[1] **Haiyan Zhao**[1*] **Shuo Wang**[1] **Hanqing Wang**[2] **Zhiyuan Liu**[1]

[1]Tsinghua University
[2]Shanghai University of Finance and Economics
wang-xj22@mails.tsinghua.edu.cn
zhao_haiyan@foxmail.com

Abstract

Parameter-Efficient Fine-Tuning (PEFT) methods such as LoRA have significantly improved the adaptation of LLMs to downstream tasks in a resource-efficient manner. However, in multi-task scenarios, challenges such as training imbalance and the seesaw effect frequently emerge. Mixture-of-LoRA (MoLoRA), which combines LoRA with sparse Mixture-of-Experts, mitigates some of these issues by promoting task-specific learning among experts. Despite this, MoLoRA remains inefficient in terms of training speed, parameter utilization, and overall multi-task performance. In this paper, we propose **M**ixture of **A**symmetric **Lo**w-**R**ank **A**daptaion (MALoRA), a flexible fine-tuning framework that leverages asymmetric optimization among LoRA experts. MALoRA reduces the number of trainable parameters by 30% to 48%, increases training speed by 1.2x, and matches the computational efficiency of single-task LoRA models. Additionally, MALoRA addresses overfitting issues commonly seen in high-rank configurations, enhancing performance stability. Extensive experiments across diverse multi-task learning scenarios demonstrate that MALoRA consistently outperforms all baseline methods in both inter-domain and intra-domain tasks.

1 Introduction

Large Language Models (LLMs), such as BLOOM (Le Scao et al., 2023), LLaMA (Touvron et al., 2023a,b), and Mixtral 8x7B (Jiang et al., 2023, 2024) have demonstrated remarkable general capabilities. These models, pre-trained on large and diverse datasets, can be adapted to new tasks through fine-tuning, leading to state-of-the-art performance in downstream applications (Chung et al., 2024; Wei et al., 2021), which extend their applicability across varied domains, significantly broadening their scope of use.

Fine-tuning LLMs by updating all parameters is computationally expensive and resource-intensive. To address this, PEFT methods such as Adapter (Houlsby et al., 2019), LoRA (Hu et al., 2021) and DoRA (Liu et al., 2024b) were proposed. These methods focus on fine-tuning smaller, localized modules, significantly reducing memory usage and communication overhead. Despite their success, these methods still face limitations in complex multi-task scenarios.

Recent studies have highlighted several limitations of LoRA, such as training imbalances, catastrophic forgetting, and poor generalization to unseen tasks (Liu et al., 2024a; Luo et al., 2024; Liu and Luo, 2024). To address these issues, Mixture-of-LoRA (MoLoRA) was introduced, combining LoRA with a sparse Mixture-of-Experts (MoE) architecture. MoLoRA mitigates the training imbalances of LoRA by distributing tasks across multiple experts, allowing each expert to perform specialized functions and reducing the risk of catastrophic forgetting. The MoE router dynamically assigns weight coefficients to each LoRA expert, improving both task specialization and overall generalization in multi-task learning (Zadouri et al., 2023).

However, MoLoRA introduces additional challenges, such as increased training latency and parameter redundancy. Existing work related to MoLoRA primarily focus on improving its routing strategies (Dou et al., 2024; Liu and Luo, 2024; Li et al., 2024a) , yet they often overlook the intrinsic relationships and redundancies among LoRA experts. Inspired by the observed asymmetries in LoRA (Zhu et al., 2024), we conducted experiments to analyze the similarities and asymmetries between MoLoRA experts. Our findings reveal significant parameter redundancy, particularly in the down-projection matrices, where different experts show a high degree of similarity, indicating inefficiencies in its parameter usage. In contrast, the up-projection matrices exhibit much

*Corresponding author.

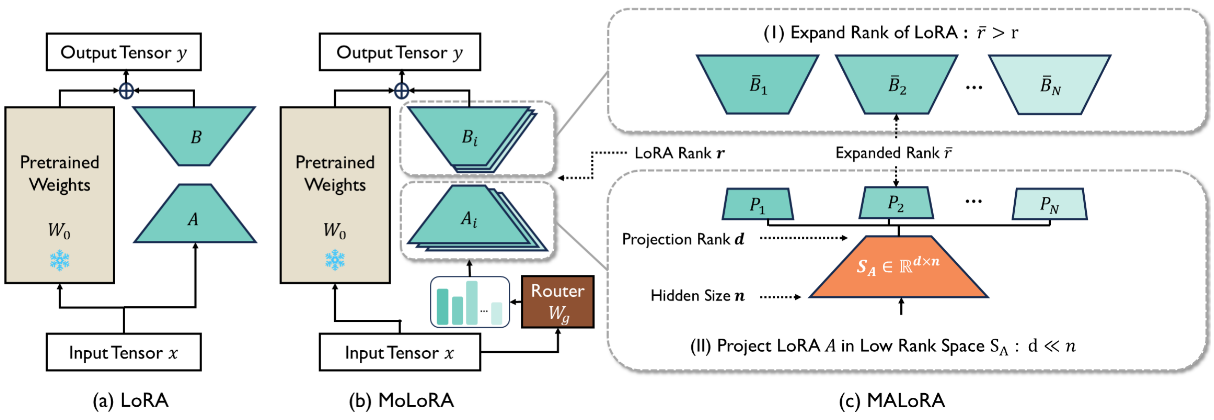


Figure 1: Architectures Overview: (a) LoRA, (b) MoLoRA, and (c) the proposed MALoRA. MALoRA optimizes the model in two key ways: (I) it increases the rank of the up-projection matrices (B_t) to enhance expert generalization capabilities, and (II) it introduces a shared low-rank subspace (S_A) in the down-projection matrices (A_t), while assigning each expert a unique coefficient matrix (P_t), effectively reducing parameter redundancy and computation.

less similarity, suggesting the need for additional capacity to better capture task-specific variations.

To address these inefficiencies, we propose Mixture of Asymmetric Low-Rank Adaptation (MALoRA), as shown in Figure 1. MALoRA introduces a shared, tunable low-rank subspace in the down-projection module, with each LoRA expert assigned a compact coefficient matrix, effectively reducing parameter count and computational complexity while preserving distinctions between experts. By reallocating the parameters saved from the down-projection module to the up-projection module, MALoRA increases the rank of the up-projection matrices, enhancing the theoretical generalization bounds of the model. By leveraging these asymmetries, MALoRA optimizes parameter usage, reduces redundancy, and improves both generalization and multi-task learning performance.

In summary, our contributions are as follows:

(1) We identify and quantify the parameter redundancy in MoLoRA, particularly in the down-projection matrices, where experts demonstrate significant overlap. This insight highlights inefficiencies in the current parameter allocation strategies, with distinct optimization needs identified for the up- and down-projection modules.

(2) We propose MALoRA, a novel fine-tuning framework that introduces a shared low-rank subspace in the down-projection module, reducing redundancy and computational overhead. The reallocated parameters are utilized to enhance the up-projection module, expanding its rank, and improving the model's theoretical generalization capacity.

(3) MALoRA reduces the number of trainable parameters by 30% - 48%, while maintaining or surpassing the performance of MoLoRA across multi-tasks. It further achieves a 1.2x speedup in training and inference, alleviating high-rank overfitting and optimizing computational efficiency.

(4) We conduct extensive evaluations across both inter-domain and intra-domain multi-task learning scenarios, demonstrating that MALoRA consistently outperforms all baseline methods in both efficiency and generalization, offering a scalable and robust solution for fine-tuning large models.

2 Preliminaries

LoRA The Low-Rank Adaption (Hu et al., 2021) assumes that the updates to the linear weight $W \in \mathbb{R}^{m \times n}$ exhibit a low-rank structure. It employs two trainable low-rank matrices $A \in \mathbb{R}^{r \times n}$ and $B \in \mathbb{R}^{m \times r}$ to approximate the parameter update of W during fine-tuning:

$$\Delta W = \frac{\alpha}{r} BA \tag{1}$$

Here, r represents the rank of decomposed matrices and α controls the scale of the update. Given that $r \ll \min\{n, m\}$, LoRA greatly reduces memory and GPU communication overhead.

AsyLoRA Asymmetry LoRA (Zhu et al., 2024) builds on LoRA by leveraging the inherent asymmetry between the low-rank matrices A and B. Specifically, A tends to extract features from the input while B refines these features to align with task-specific objectives.

AsyLoRA introduces an asymmetric resource allocation strategy. It focuses computational budgets on the matrix B by doubling the rank of LoRA, while keeping it fully trainable and freezing the parameters of matrix A, that is,

$$\Delta W = \frac{\alpha}{r} B^{m\times 2r} A^{2r\times n} \tag{2}$$

The work brought forward the generalization boundary for LoRA methods. Given a distribution μ and a trainable parameter set $\mathcal{A}$, is reformulated as follows. Here, γ is a constant influenced by training hyperparameters, and $|\mathcal{A}|$ represents the number of trainable parameters:

$$|gen(\mu, \mathcal{A})| \leq \sqrt{\gamma|\mathcal{A}|} \tag{3}$$

Mixture-of-LoRA Mixture-of-LoRA (Zadouri et al., 2023) (MoLoRA) is a novel PEFT framework that integrates the MoE architecture with LoRA modules serving as experts. The built-in router of MoE dynamically distributes input data to different LoRA experts. Specifically, a MoLoRA layer consists of N independent LoRA experts $\{E_t\}_{t=1}^{N}$. The router, parameterized by a learnable weight matrix W_g, assigns routing weights G_t to each expert t based on the input tensor x. The routing decision and the layer output are computed as follows:

$$\Delta W_t = B_t A_t \tag{4}$$

$$G_t = \text{TopK}(\text{Softmax}(W_g \cdot x)) \tag{5}$$

$$y = \sum_{i=t}^{N} G_t \Delta W_t x \tag{6}$$

where G_t represents the top K experts selected by the router, ensuring that only the most relevant experts are activated. This dynamic allocation allows MoLoRA to efficiently handle multi-task scenarios by assigning appropriate experts to each task. MoLoRA has demonstrated strong generalization to unseen tasks and excels in multi-task learning settings (Luo et al., 2024; Li et al., 2024a).

3 Method

We propose MALoRA, which is built on the observed asymmetries of LoRA experts within the MoLoRA architecture. MALoRA improves parameter efficiency and enhances generalization in multi-task learning by leveraging both the similarities and asymmetries in the down-projection (A) and up-projection (B) matrices. This approach reduces the number of trainable parameters by at least 30% and reallocates the saved resources to expand generalization capacity of the model, all without adding significant computational overhead.

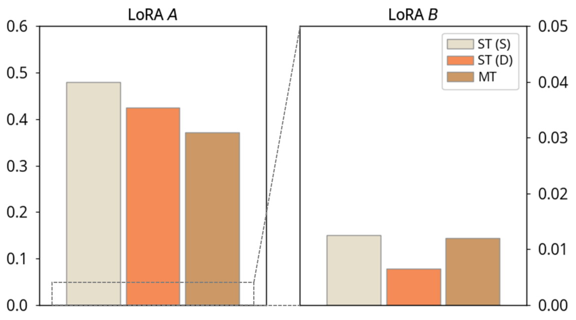


Figure 2: Spatial Similarity Analysis. Spatial similarity between LoRA experts within the same MoLoRA layer, evaluated using CCA. The down-projection matrix (A) demonstrates significantly higher similarity across all learning scenarios (ST(S), ST(D), MT), suggesting it captures generalized features. In contrast, the up-projection matrix (B) shows much lower similarity, indicating its role in task-specific fine-tuning.

3.1 Asymmetry in MoLoRA

To explore the inter-expert relationships in MoLoRA, we analyzed the spatial similarity between the down-projection (A) and up-projection (B) matrices across different tasks. As illustrated in Figure 2, the down-projection matrices (A) exhibit significantly higher similarity compared to the up-projection matrices (B), indicating a notable asymmetry in their behavior.

We performed a Canonical Correlation Analysis (CCA) (Ramsay et al., 1984) to evaluate the similarities between LoRA experts under three different learning scenarios: same task (ST(S)), different tasks (ST(D)), and multi-task learning (MT). The results show that A consistently maintains higher similarity across all scenarios. As task complexity increases from ST(S) to MT, the similarity of A decreases, reflecting its adaptability to more diverse task settings. Despite this decline, A still retains a relatively high similarity, indicating that it captures generalized patterns while adapting to task-specific nuances by marginal differences. In contrast, B demonstrates much lower similarity, with values below 0.05 across all scenarios, suggesting that it is more focused on task-specific fine-tuning rather than generalization.

The singular value distribution in Figure 3 illustrates the different roles of the down-projection (A) and up-projection (B) matrices. For B, the singular values are closely clustered within a narrow range between 0.75 and 1.25, indicating that most basis vectors contribute similarly and are uniformly important for task-specific fine-tuning. In contrast, A exhibits a wider range of singular values, spanning

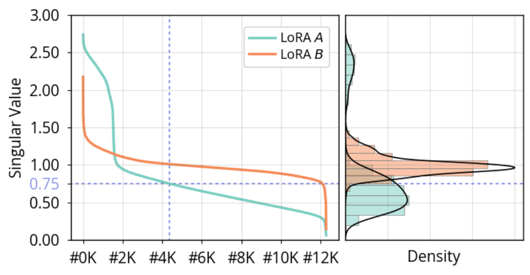


Figure 3: Singular Values of the Concatenated Homologous Matrices in descending order. matrix B shows a concentration of larger singular values, indicating that many singular vectors are important for task-specific fine-tuning. In contrast, A has more smaller singular values, with only a few larger ones, suggesting that only a subset of singular vectors play a critical role. This reflects that B distributes importance across more components, while A relies on a smaller, more focused set of key features for generalization.

from 0.25 to 2.75. This wider distribution suggests that a smaller subset of basis vectors with larger singular values (approximately 33% of the total) plays a dominant role in capturing key features. The remaining vectors, associated with smaller singular values, contribute less significantly. This indicates that A can be effectively adapted to tasks by focusing on vectors with larger singular values, allowing the model to capture generalizable and task-specific information more efficiently.

3.2 Projecting A in Low-Rank Space

Given the observed similarities among the down-projection matrices A_t across different LoRA experts, MALoRA optimizes A_t by projecting it into a layer-shared low-rank subspace. This subspace is defined by a matrix $S_A \in \mathbb{R}^{d\times n}$, which is shared among all experts in a MoLoRA layer. This allows for more efficient feature extraction while reducing redundancy among experts.

Instead of treating each expert's down-projection matrix independently, MALoRA projects each A_t into the row space of S_A, capturing the most relevant features in a lower-dimensional space. It is verified in the previous section, that the model only needs to learn the basis vectors with the larger singular values, which constitute 33% of the total. Thus, the rank d of S_A is kept small relative to the original matrix dimensions, enabling efficient compression of A_t without significant loss of expressiveness. A_t for each expert is then replaced by a projection coefficient matrix $P_t \in \mathbb{R}^{r\times d}$, which adjusts the contributions of the shared S_A basis vectors for each expert to better align with task-specific needs.

Mathematically, for a MoLoRA layer with N experts, MALoRA redefines the parameter set as $\{W_g, S_A\} \cup \{(P_t, \overline{B}_t)\}_{t=1}^{N}$, where $\overline{B}_t$ is the expanded up-projection matrix (details provided in the next subsection). During forward propagation, A_t is replaced by its projection into span(S_A), leading to the following expressions:

$$\Pi_{\text{span}(S_A)}(A_t) = P_t S_A \tag{7}$$

$$\Delta W_t = \overline{B}_t P_t S_A \tag{8}$$

For initialization, the up-projection matrix $\overline{B}_t$ is set to zero, consistent with the LoRA approach. While the down-projection matrices in LoRA are initialized using the Kaiming Uniform initialization (He et al., 2015) to ensure proper gradient scaling, we aim for the product $P_t S_A$ to similarly follow a Kaiming Uniform distribution as well. To achieve this, we apply singular value decomposition (SVD) to several Kaiming Uniform-initialized random matrices $K_t \in \mathbb{R}^{d\times n}, t \in [0, N]$, where each expert receives its own initialized matrix. Since K_t has rank d ($d < n$), the singular vectors beyond rank d are zero-filled and cropped from the SVD results, as shown in Eq. 9.

The product of U_t and Σ_t is clipped to rank r for the initialization of P_t, S_A is initialized using the right singular matrix V_0 from the first SVD decomposition (K_0), as it serves as a common basis across all experts within the layer. This initialization establishes a global subspace shared by all experts, while P_t is responsible for capturing task-specific adaptations. The initialization is given by:

$$(U_t)_{d\times d}, (\Sigma_t)_{d\times d}, (V_t)_{d\times n} = \text{SVD}_{\text{crop}}(K_t) \tag{9}$$

$$P_t = (U_t\Sigma_t)[:\text{r}]/\beta \tag{10}$$

$$S_A = \beta V_0 \tag{11}$$

where β is a hyperparameter that controls the balance between general and task-specific learning. A higher β amplifies the gradients of P_t, thus enhancing the model's ability to focus on task-specific details. In contrast, a lower β shifts the emphasis toward capturing generalized features by giving more weight to S_A. Please refer to Appendix A.4 and A.8.3 for detailed analysis and guidance.

3.3 Expanding Rank of LoRA Experts

In Section 3.1, we discuss reducing the dimensionality of the down-projection matrices A_t. By setting the rank of the shared matrix S_A to

$d = \lambda r N$, where $\lambda < 1$. r is the rank of each LoRA expert and N is the number of experts, we eliminate the need for separate down-projection matrices. This reduction reallocates resources to expand the up-projection matrix B, increasing its rank to $\overline{r} = r + (1-\lambda)r$ for each LoRA expert. In practice, λ is recommended as a value of $\frac{1}{2}$, which derives $\frac{d}{N\overline{r}} = \frac{\lambda N r}{N\overline{r}} = 33\%$ aligning with Figure 3.

The computational savings from compressing A_t are reallocated to expand the up-projection matrices $\overline{B}_t$. The newly introduced matrix P_t, sized only $\overline{r} \times d$, is relatively small compared to the hidden size n, contributing less than 1% to the total parameter count of PEFT. Thus, the additional parameters introduced by MALoRA are negligible. Expanding the rank of LoRA experts enables MALoRA to enhance the model's generalization capability. According to the generalization bound derived in Section 2, the upper bound $\mathcal{U}$ is given by:

$$\mathcal{U}(|gen(\mu, \mathcal{A})|) \propto \sqrt{|\mathcal{A}|} \propto \sqrt{r} \qquad (12)$$

For each expert, since the ranks of S_A, P_t, and $\overline{B}_t$ all exceed $\overline{r}$, the resulting rank of ΔW_t is $\overline{r}$, whereas MoLoRA experts retain the original rank r. MALoRA establishes a generalization boundary that is $\sqrt{\overline{r}/r}$ times higher than that of MoLoRA, resulting in improved performance across tasks.

4 Experiments

In this section, we evaluate MALoRA's performance across diverse multi-task learning scenarios. We describe the datasets used for training and evaluation, outline the implementation details, and compare MALoRA with baseline methods. The results highlight MALoRA's efficiency and accuracy improvements, followed by an ablation study to examine the impact of key components.

4.1 Experimental Setup

Datasets To evaluate MALoRA, we conducted experiments in both inter-domain and intra-domain settings. We selected datasets from different domains, including MetaMathQA, Magicoder, MedMCQA, and Finance Alpaca, representing tasks such as math reasoning, code generation, medical knowledge, and finance, respectively. Additionally, we included the E2E dataset for specialized task evaluation and Alpaca-GPT4 to test common-sense reasoning and instruction-following. The multi-domain training set was created by sampling 30,000 instances from each dataset and blending them uniformly. For evaluation, we used GSM8K for math, HumanEval for code, Financial PhraseBank for finance, and ARC for common-sense reasoning, along with task-specific test splits for other domains.

For intra-domain evaluation, we focused on multi-task learning within the domain of common-sense reasoning. We employed datasets like PIQA, OBQA, BoolQ, and ARC to evaluate MALoRA's performance on closely related tasks. This allows us to measure generalization capacities within a specific domain, assessing how effectively it handles tasks with similar data distributions. Details of datasets can be found in Appendix A.1.

Implementation Details We adopt LLaMA-2 7B as the backbone model and compare MALoRA against LoRA, DoRA, Asymmetry LoRA, MoLA, and MoLoRA. We also combines Asymmetry LoRA with routers to form MoAsyLoRA. For both LoRA and DoRA, the rank r is set to 64, with a dropout rate of 0.05. Asymmetry LoRA doubles the rank, keeping the matrix A frozen during fine-tuning. Following mainstream practices, MoLoRA uses 8 LoRA experts, each with a rank of $r = 8$, and activates the top-2 experts based on input routing, with an auxiliary loss factor of 0.001. MoLA assigns 2,4,6, and 8 experts to layers from low to high. MALoRA expands the rank to $\overline{r} = 12$ according to $\lambda = \frac{1}{2}$, and the shared matrix S_A has a rank of $d = 32$, ensuring parameter comparability across all baselines. Additionally, MALoRA-Small is introduced, where $\overline{r} = 8$ and $d = 22$, providing a more parameter-efficient variant for comparison.

The PEFT modules are applied to all linear layers within the transformer architecture. All backbone parameters remain frozen throughout the experiments. For inter-domain learning tasks, the learning rate is set to $5e^{-4}$ with a batch size of 4, while intra-domain tasks, the learning rate is $4e^{-4}$ with a batch size of 2. These hyperparameters were determined via a grid search to optimize performance across tasks. Further details can be found in Appendix A.3.

4.1.1 Main Result

Inter-domain Performance As shown in Table 1, MALoRA outperforms all baselines in inter-domain settings, achieving an average score of 56.3, surpassing both LoRA and MoLoRA by 1.6% and 1.0% respectively. This performance is particularly notable on tasks like GSM8K and HumanEval,

Method	**#Params**	**Latency(μs)**	**MedMCQA**	**HumanEval**	**GSM8K**	**PhraseBank**	**ARC-C**	**ARC-E**	**E2E**	**AVG.**
LLaMA-2 7B	-	-	34.2	14.6	13.2	57.6	43.1	75.5	21.7	37.1
LoRA (ST)	2.1%	-	42.5	34.8	36.5	77.4	62.4	79.3	66.4	57.0
AsyLoRA (ST)	2.2%		42.6	30.5	33.1	78.8	61.2	77.4	66.0	55.7
LoRA	2.1%	**833.3**	43.2	**34.8**	34.0	69.6	59.1	76.5	65.9	54.7
AsyLoRA	2.2%	-	**43.3**	31.1	27.6	74.2	59.8	77.8	65.5	54.2
DoRA	2.1%	1020.9	41.3	32.3	32.6	74.0	60.0	75.6	**66.5**	54.6
MoAsyLoRA	2.3%	-	40.6	24.4	23.9	73.0	58.5	76.0	65.8	51.7
MoLoRA	2.2%	1072.3	42.1	29.3	33.1	78.1	**61.1**	78.1	65.6	55.3
MoLA	**1.5%**	-	41.0	24.4	33.4	76.1	59.1	77.9	66.3	54.1
MALoRA-Small	1.6%	896.4	41.0	29.9	34.0	**80.3**	59.4	77.2	65.7	55.4
MALoRA	2.3%		42.3	32.9	**34.1**	79.8	60.0	**78.5**	66.3	**56.3**

Table 1: Comparison of various PEFT methods for multi-task learning across different domains. "ST" stands for single downstream task fine-tuning. For the E2E task, the ROUGE-L metric is used, while accuracy is reported for others. "#Params" refers to the percentage of trainable parameters relative to the base model. MALoRA and MALoRA-Small achieve the best results with the lowest latency, highlighting their efficiency.

Method	**PIQA**	**OBQA**	**BoolQ**	**ARC-C**	**ARC-E**	**AVG.**
LoRA	81.1	89.4	77.1	64.5	80.5	78.48
AsyLoRA	80.4	87.4	**82.0**	64.9	82.4	79.42
DoRA	80.0	89.2	80.4	65.9	82.4	79.56
MoLoRA	80.3	**90.4**	79.3	66.1	82.4	79.69
MoLA	81.1	87.6	79.4	**66.8**	81.9	79.35
MALoRA-Small	81.6	87.8	81.8	66.6	82.6	80.08
MALoRA	**81.8**	89.6	81.6	66.6	**82.8**	**80.47**

Table 2: Comparison of different PEFT methods on intra-domain common-sense reasoning tasks. MALoRA achieves the highest average score, highlighting its effectiveness in handling tasks within the same domain.

where MALoRA demonstrates its ability to handle task complexity. These gains are attributed to MALoRA's effective integration of an asymmetric low-rank adaptation and multi-expert MoE structure, which extends the generalization boundary of the model while maintaining a low computational overhead. Baselines without an MoE structure struggled on tasks like Finance PhraseBank due to data imbalance, where label-heavy tasks like Magicoder dominated, leading to underfitting in minority datasets such as financial data. This highlights the seesaw phenomenon often seen in traditional PEFT methods. In contrast, MALoRA's ability to handle diverse multi-domain tasks showcases its robustness in orthogonal domain learning with significant distribution differences. MALoRA-Small, with only 1.6% trainable parameters, uses fewer parameters than any other method while achieving an impressive average performance. Although its overall results are slightly lower than MALoRA, MALoRA-Small still surpasses MoLA by 1.3%, which has comparable additional parameter amounts, and outperforms all other multi-task methods. This result emphasizes MALoRA's ability to scale down without sacrificing much performance, making it an ideal solution for scenarios where computational efficiency is a priority. More analysis are available in Appendix A.1.3 and A.5.

Additionally, the latency results clearly demonstrate MALoRA's efficiency. It reduces training time by 16% compared to MoLoRA, while achieving higher performance on most tasks. Notably, MALoRA's latency is comparable to LoRA's, despite incorporating a more complex multi-expert structure. This shows that MALoRA can offer significant performance improvements without incurring a substantial increase in computational cost.

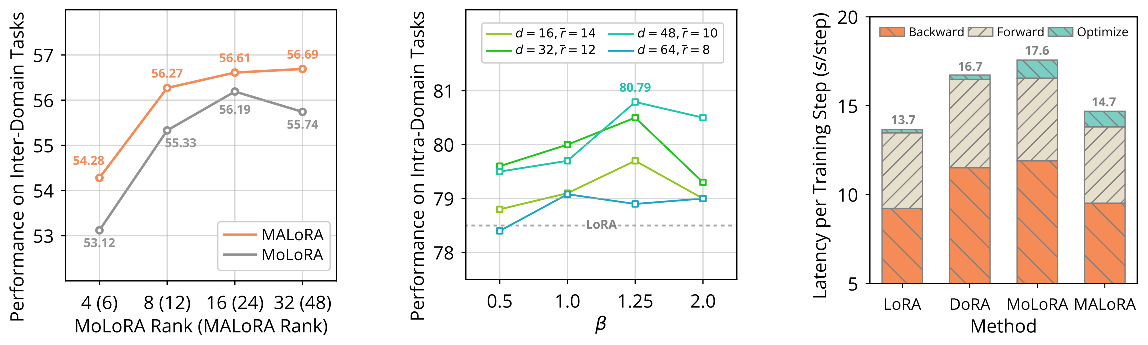
Intra-domain Performance For common-sense reasoning tasks, as shown in Table 2, MALoRA again demonstrates its superiority. It improves upon LoRA by 2% and MoLoRA by 0.8%, proving its efficacy even in more focused intra-domain tasks. MALoRA-Small consistently surpasses baselines while reducing 30% trainable parameters. The robust performance of MALoRA on various tasks indicates that it strikes a good balance between learning domain-specific features and maintaining generalizability. It also highlights MALoRA's ability to manage data imbalance, as seen in tasks like ARC-C, where traditional methods like LoRA struggle.

4.2 Ablation Study

To gain deeper insight on the contribution of each component in MALoRA, the results of the ablation study are presented in Table 3. Specifically, we compare MALoRA's performance with various modifications to assess the impact of asymmetry, shared subspaces, and rank configurations. For

Method	**#Params**	**MedMCQA**	**HumanEval**	**GSM8K**	**PhraseBank**	**ARC-C**	**ARC-E**	**E2E**	**AVG.**
MoLoRA	2.2%	42.1	29.3	33.1	78.1	61.1	78.1	65.6	55.3
MoLoRA, $r = 12$	**3.3%**	40.8	31.7	**34.9**	77.3	**62.6**	78.2	66.2	56.0
w same A_t, $r = 12$	**3.3%**	42.4	29.9	33.7	79.1	61.9	**78.8**	66.3	56.0
MALoRA	2.3%	42.3	**32.9**	34.1	**79.8**	60.0	78.5	**66.3**	**56.3**
w fixed S_A, $r = 16$	2.4%	43.2	28.7	32.7	73.9	58.6	76.8	66.0	54.3
w/o Asymmetry	2.2%	42.4	29.3	35.4	**79.8**	59.6	77.8	65.9	55.8
w/o shared S_A	2.3%	**43.9**	28.1	29.7	66.2	61.1	78.1	65.8	53.3
w/o ft P_t	2.3%	41.9	29.9	34.1	78.9	58.7	78.8	65.8	55.5
decomposing B_t	2.2%	43.0	30.5	32.1	74.1	61.3	76.6	65.9	54.8

Table 3: Ablation study results for MALoRA, showing the impact of different components on model performance.



(a) Performance Across Ranks (b) Ablation Study of β and d (c) Training Latency Comparison

Figure 4: (a) Multi-domain learning performance across different ranks, with methods maintaining a comparable number of trainable parameters on the same x-axis. (b) Ablation study of hyperparameters β and d in common-sense multi-task learning. (c) Comparison of training latency for various PEFT methods with FastMoE.

MoLoRA, initializing matrices A_t in the same layer with a fixed random matrix shows a comparable performance against MoLoRA, suggesting that initializing experts similarly does not significantly damage multi-task learning performance. Even when experts in MALoRA are projected into an identical low-rank space span(S_A), the model outperforms MoLoRA while reducing trainable parameters by 30%, confirming the compressibility of MoLoRA.

For MALoRA variants, *w fixed* S_A, which freezes S_A and expands $\overline{r}$ to 16, leads to performance degradation. This implies that a randomized S_A that is not updated during training fails to capture meaningful task-specific features. The lack of adaptability prevents the model from aligning the subspace with task distributions.

w/o Asymmetry, which uses a symmetric structure of MALoRA with LoRA rank $r = 8$ and $d = 64$, shows that increasing LoRA rank is more effective in improving learning capacity than expanding d. Even with the same number of trainable parameters, the asymmetric structure in MALoRA extends the generalization boundary. In contrast, *w/o shared* S_A that replaces the shared subspace S_A with distinct down-projection matrices for each expert limits the rank of each expert's matrix to $\frac{d}{N}$. This reduced rank constrains the generalization boundary and leads to degraded performance.

Freezing P_t without fine-tuning leads to a notable performance decline, as it prevents the model from fully leveraging the low-rank space S_A and limits its ability to adapt to task-specific features. When decomposition is applied to B_t instead of A_t in *decomposing* B_t, performance decreases by 1.5% on average. This compression of B_t reduces the expressive capacity of the model, as B_t has a lower inter-expert similarity than A_t, highlighting the need for more adaptable representations in B_t to handle task-specific variations effectively.

Rank Robustness We investigates the effectiveness of MALoRA in various ranks $\overline{r}$. As shown in Figure 4(a), MALoRA consistently outperforms MoLoRA in all rank configurations, achieving significant improvements of up to 1.16% at rank 4(6) (the values in parentheses represent the rank of MALoRA) and maintaining a clear advantage throughout. Most notably, MALoRA at rank $\overline{r} = 12$ matches the performance of MoLoRA at a higher rank of $r = 16$, while reducing trainable parameters by 48%. This result underscores MALoRA's ability to maintain high performance with fewer resources. The diminishing returns in MoLoRA as ranks increase indicate that simply increasing rank without efficient adaptation leads to overfitting at higher ranks. In contrast, MALoRA scales its capacity more effectively as rank increases. This is likely because its asymmetric structure focuses on fine-tuning the projection coefficients P_t for task-specific adaptation, while maintaining the stability of the shared subspace S_A, helping to prevent overfitting in high-ranks.

Hyperparameter Robustness We conducted experiments to evaluate MALoRA's sensitivity to hyperparameters β and d, with changes in d also adjusting $\overline{r}$ to maintain consistent trainable parameters. As shown in Figure 4(b), extreme values of d—too large ($d = 64, \overline{r} = 8$) or too small ($d = 16, \overline{r} = 14$)—lead to suboptimal results, emphasizing the need for a balanced allocation between the shared subspace and expert-specific components. The best performance occurs at $d = 48, \overline{r} = 10$, achieving a balance between shared and task-specific representations. MALoRA performs optimally when $\beta = 1.25$, striking the ideal balance between refining task-specific features in P_t and maintaining the stability of S_A. When $\beta > 1$, the model places more focus on task-specific learning, enhancing performance in scenarios that require stronger task differentiation. However, Figure 4(b) and additional results under different task combinations in A.8.2 shows that MALoRA outperforms other baselines in the vast majority of cases.

Latency As shown in Figure 4(c), MALoRA reduces latency (second/step) by 1.2x compared to MoLoRA during a single training step. By projecting inputs into the low-rank space span(S_A), MALoRA decreases the dimensionality in each MoE layer, reducing the latency of gradient computations, data routing, and result collection. This results in significantly faster backward propagation due to more efficient gradient calculations in the compressed low-rank space. Although forward computations still have some complexity, the reduction in communication time compensates, leading to an overall time decrease per step. When integrated with FastMoE (He et al., 2021), both LoRA and MALoRA complete a training step in 4.3 seconds, while MoLoRA and DoRA take 4.65 and 4.97 seconds, respectively. The modular structure introduces a slight increase in optimization time, but the overall latency reduction achieved by MALoRA remains substantial. Ultimately, MALoRA's efficient use of low-rank projections and modular design significantly reduces forward and backward propagation times, improving training speed without sacrificing performance. The details of the latency are provided in Appendix A.6.

5 Discussion

5.1 Routing Distribution Analysis

To further explore the inner working mechanism of MALoRA, we performed an interpretability analysis of the routing distribution in different tasks. The results are shown in Figure 5. Differentiated by routing, the functionality of the experts in MALoRA exhibits clear differentiation, while different types of tasks activate unique experts, and similar tasks may share experts (e.g., ARC-C and ARC-E). It indicates that MALoRA is capable of recognizing tasks and mapping them to the appropriate expert.

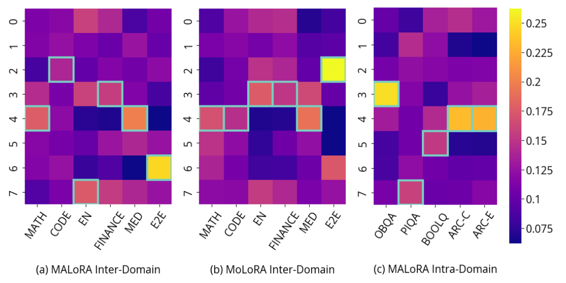


Figure 5: Expert Routing Distribution. Figure presents the routing states in (a) MALoRA for Inter-Domain Tasks (b) MoLoRA for Inter-Domain Tasks and (c) MALoRA for Intra-Domain Tasks. The most important experts for each task, identified as those with the highest assigned weight, are highlighted with boxes.

Remarkably, MALoRA emerges with comprehensive selection and utilization of experts compared to MoLoRA. Observing the expert with the highest weight for each task, MoLoRA tends to rely more concentrically on expert 2,3,4 in Figure 5(b), while there is no phenomenon of expert degeneration or abandonment in MALoRA. This phenomenon may be attributed to the fact that in MALoRA, the matrix S_A projects experts into a designated low-rank space, enabling different experts to jointly utilize overlapping dimensions. This mechanism facilitates cross-expert knowledge transfer and mitigates the winner-takes-all phenomenon.

5.2 Expert Allocation Comparison

MALoRA decomposes experts within a shared low-rank subspace, reducing redundancy while preserving functionality. However, it remains unclear whether this decomposition affects the role of LoRA experts in MoLoRA. Figure 6 presents the cosine similarity between expert pairs in MALoRA and MoLoRA, initialized with the same weights.

The high similarity along the main diagonal indicates that expert functions remain largely consistent, suggesting that MALoRA effectively preserves task-specific learning while optimizing parameter efficiency.

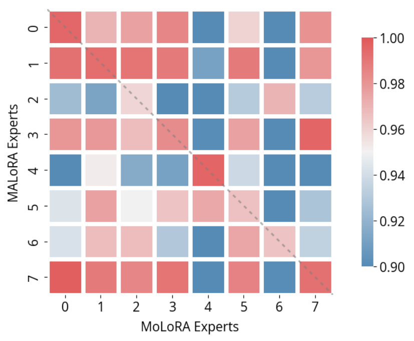


Figure 6: Pairwise Cosine Similarities of Experts' Routing Distributions in MALoRA and MoLoRA. The Diagonal of the matrix are mainly in authority, illustrating alignments of experts. Due to the concentration of routing weights near the average value, the similarities are generally large, but this does not hinder the analysis.

6 Related Work

6.1 Redundancy of PEFT Methods

PEFT methods such as Prefix-Tuning (Li and Liang, 2021), Prompt-Tuning (Lester et al., 2021), and LoRA (Hu et al., 2021) are widely adopted to reduce the computational cost of fine-tuning LLMs. Recent research has highlighted two main areas for improvement in LoRA: structural inequity and redundancy. To address inequity, methods like LoRA+ (Hayou et al., 2024), Asymmetry LoRA (Zhu et al., 2024), and LoRA-FA (Zhang et al., 2023) adopt unequal fine-tuning strategy of LoRA modules. On the redundancy front, VeRA (Kopiczko et al., 2023) and VB-LoRA (Li et al., 2024b) reduce the number of trainable parameters by sharing parameters across LoRA. MoLA (Gao et al., 2024) explores the optimal distribution of LoRA experts in MoLoRA, suggesting that higher layers benefit from a greater number of experts.

6.2 Mixture-of-Experts

MoE (Jacobs et al., 1991) distributes data to sub-modules for processing via a routing mechanism, activating the neurons of the feed-forward sub-layers through sparse parameters. SparseMoE (Shazeer et al., 2017) and GShard (Lepikhin et al., 2020) substantially expand the learning capacity of Transformers by incorporating numerous FFN experts within MoE framework. LLMs based on MoE architectures, such as Mixtral 8x7B (Jiang et al., 2024), have demonstrated remarkable performance. Hybrid models that integrate LoRA as experts aim to improve model capacity and better fit complex data during fine-tuning, paralleling the approach of MoE models. Relevant studies (Zadouri et al., 2023; Wu et al., 2024) explore the potential of using LoRA as parameter-efficient experts across NLP and Vision application scenarios. MoRAL (Yang et al., 2024) leverages MoLoRA to enhance lifelong learning. Approaches like MoELoRA and MixLoRA (Luo et al., 2024; Li et al., 2024a) employ MoLoRA to mitigate catastrophic forgetting in multi-task learning. The effectiveness of MoLoRA has been broadly validated, solidifying it as a key trend in PEFT research.

7 Conclusion

MALoRA introduces a novel PEFT approach for multi-task learning by leveraging a MoE structure with asymmetric low-rank adaptation. Through a shared down-projection space and expanded up-projection ranks, MALoRA optimizes parameter efficiency while enhancing generalization across tasks. Our experiments demonstrate that MALoRA consistently outperforms existing methods like LoRA and MoLoRA, particularly excelling in complex inter-domain and intra-domain scenarios. By reducing trainable parameters by up to 48% and achieving a 1.2x increase in training speed, MALoRA addresses key challenges such as training imbalances and overfitting phenomena observed in previous methods, offering a robust, scalable solution for diverse applications in multi-task learning.

8 Limitation

The proposed MALoRA method outperforms previous PEFT baselines while effectively reducing both the training parameters and the latency. However, to ensure that the enhancement of multi-task learning capability remains independent of the gating network, our approach maintains strict consistency in routing strategy and hyperparameter settings of MoLoRA. The optimization of MALoRA is network-agnostic, suggesting that customized routing strategies could further improve its performance, and it is not involved in our research. We leave this exploration for future work. Moreover, multi-task fine-tuning may trigger data security and privacy issues in certain application scenarios. More research on LLM safety is needed.

References

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

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

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

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 *NAACL*.

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:1803.05457v1*.

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

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, et al. 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.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the State-of-the-Art of End-to-End Natural Language Generation: The E2E NLG Challenge. *Computer Speech & Language*, 59:123–156.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. 2024. Higher layers need more lora experts. *arXiv preprint arXiv:2402.08562*.

Gaurang Bharti. 2024. finance-alpaca (revision 51d16b6).

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

Chaoqun He, Renjie Luo, Shengding Hu, Yuanqian Zhao, Jie Zhou, Hanghao Wu, Jiajie Zhang, Xu Han, Zhiyuan Liu, and Maosong Sun. 2024. Ultraeval: A lightweight platform for flexible and comprehensive evaluation for llms. *Preprint*, arXiv:2404.07584.

Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. 2021. Fastmoe: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262*.

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.

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

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

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki Markus Asano. 2023. Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

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

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

Yang Li, Shaobo Han, and Shihao Ji. 2024b. Vb-lora: Extreme parameter efficient fine-tuning with vector banks. *arXiv preprint arXiv:2405.15179*.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2024a. When moe meets llms: Parameter efficient fine-tuning for multi-task medical applications. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1104–1114.

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

Zefang Liu and Jiahua Luo. 2024. Adamole: Fine-tuning large language models with adaptive mixture of low-rank adaptation experts. *arXiv preprint arXiv:2405.00361*.

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*.

P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65.

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 *EMNLP*.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260. PMLR.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

James O Ramsay, Jos ten Berge, and George PH Styan. 1984. Matrix correlation. *Psychometrika*, 49(3):403–423.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2022. Language models are multilingual chain-of-thought reasoners. *Preprint*, arXiv:2210.03057.

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

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

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2023. Magicoder: Source code is all you need. *arXiv preprint arXiv:2312.02120*.

Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of lora experts. *arXiv preprint arXiv:2404.13628*.

Shu Yang, Muhammad Asif Ali, Cheng-Long Wang, Lijie Hu, and Di Wang. 2024. Moral: Moe augmented lora for llms' lifelong learning. *arXiv preprint arXiv:2402.11260*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. 2023. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*.

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. 2024. Asymmetry in low-rank adapters of foundation models. *arXiv preprint arXiv:2402.16842*.

A Appendix

A.1 Datasets

A.1.1 Inter-Domain Tasks Learning

Task	Domain	#Train	#Sample	Type
MedMCQA (Pal et al., 2022)	Medicine	182K		Question Answering
Magicoder (Wei et al., 2023)	Code	110K		Code Generation
Finance Alpaca (Gaurang Bharti, 2024)	Finance	69K	30,000	Instruction Following
MetaMathQA (Yu et al., 2023)	Mathematics	395K		Math Reasoning
Alpaca-GPT4 (Peng et al., 2023)	General	52K		Instruction Following
E2E NLG (Dušek et al., 2020)	End-to-End	34K		Natural Language Generation

Table 4: Description of Inter-Domain Training Datasets. We sampled 30,000 training data from each dataset to form a mixture inter-domain multi-task training set.

Task	Domain	#Test	Type
MedMCQA	Medicine	2,802[†]	Question Answering
HumanEval (Chen et al., 2021)	Code	164	Code Generation
PhraseBank (Malo et al., 2014)	Finance	3,453	Text Classification
GSM8K (Cobbe et al., 2021)	Mathematics	1,319	Math Reasoning
ARC-C (Clark et al., 2018)	Commen-Sense	1,170	Question Answering
ARC-E (Clark et al., 2018)	Commen-Sense	2,380	Question Answering
E2E NLG	End-to-End	4,693	Natural Language Generation

Table 5: Description of Inter-Domain Testing Datasets. [†] Due to the non-public nature of the labels for the MedMCQA test set, the development dataset is utilized for testing in UltraEval (He et al., 2024), and entries with anomalies in multiple-choice selections or variations in the number of options have been cleaned.

A.1.2 Intra-Domain Tasks Learning

Task	#Train	#Test	Type
ARC-C (Clark et al., 2018)	1,120	1,170	
ARC-E (Clark et al., 2018)	2,250	2,380	
PIQA (Bisk et al., 2020)	16,100	1,840	Question Answering
OpenBookQA (Mihaylov et al., 2018)	4,957	500	
BoolQ (Clark et al., 2019)	9,427	3,270	

Table 6: Description of Intra-Domain Training Datasets. Datasets focus on the scope of common-sense reasoning, following a common setting of existed works (Li et al., 2024a; Luo et al., 2024).

A.1.3 Quantities of Training Tokens

Table 7 presents the distribution of token quantities of training datasets. In the context of multi-domain learning, the Magicoder dataset contains a substantial 1,446K of label tokens for computing cross-entropy and performing backward propagation, significantly surpassing datasets in other domains. As shown in Table 1, non-MoE PEFT methods achieve performance on the HumanEval benchmark that is comparable to single-task fine-tuning. However, on the Finance Alpaca dataset, which has fewer number of label tokens, the learning efficacy of non-MoE methods is sub-optimal compared to MoE approaches. A similar trend is observed in common-sense multi-task learning, as indicated in Table 2. Due to the disparity in the magnitude of label tokens, LoRA demonstrates superior performance on the PIQA benchmark while exhibiting deficiencies on the ARC task. This highlights a clear correlation between the abundance of label tokens and LoRA's performance in the corresponding tasks. In summary, LoRA shows drawbacks related to learning imbalance and catastrophic forgetting.

Task	#Input	#Label	#Total
MedMCQA	469K	157K	626K
Magicoder	3,231K	**1,446K**	4,677K
Finance Alpaca	839K	206K	1,045K
MetaMathQA	1,415K	537K	1,952K
Alpaca-GPT4	483K	214K	697K
E2E NLG	202K	461K	663K
ARC-C	18.5K	1.8K	20.3K
ARC-E	36.4K	4.3K	40.7K
BoolQ	378.7K	18.7K	397.4K
OpenBookQA	79.8K	9.7K	89.5K
PIQA	318.1K	**31.9K**	350K

Table 7: Description of Token Quantity in Each Dataset

A.2 Initialization

MALoRA initialization process described in Section 3.2 is written as pseudo code for convenience.

Algorithm 1 MALoRA Initialization

```
Input: Number of layers $L$, expert rank $\overline{r}$, shared space rank $d$, number of experts $N$, input size $n$, output size $m$, scale factor $\beta$.
Output: MALoRA parameter set $\{W_g^l, S_A^l\} \cup \{(P_t^l, \overline{B}_t^l)\}_{t=1}^N$ for each layer $l = 1, 2, \cdots L$
 1: for $l = 1, 2, \cdots, L$ do
 2:     Initialize $W_g^l \sim \mathcal{N}(0, \frac{1}{n})$
 3:     Initialize $K_0 \in \mathcal{R}^{d\times n} \sim \mathcal{U}(-\sqrt{\frac{1}{n}}, \sqrt{\frac{1}{n}})$
 4:     $U_0, \Sigma_0, V_0 = \text{SVD\_LowRank}(K_0)$
 5:     $S_A^l = \beta V_0$
 6:     for $t = 1, 2, \cdots, N$ do
 7:         Initialize $K_t \in \mathcal{R}^{d\times n} \sim \mathcal{U}(-\sqrt{\frac{1}{n}}, \sqrt{\frac{1}{n}})$
 8:         $U_t, \Sigma_t, V_t = \text{SVD\_LowRank}(K_t)$
 9:         Initialize $\overline{B}_t^l$ by zero matrix
10:         $P_t^l = (U_t\Sigma_t)[:\overline{r}]/\beta$
11:     end for
12: end for
```

A.3 Hyperparameter Configuration

A.3.1 Inter-Domain Tasks Learning

Hyperparameters	LoRA/DoRA	AsyLoRA	MoLoRA	MALoRA
Optimizer	AdamW			
Learning Rate	5e-4			
Epochs	3			
Batch Size	4			
Dropout	0.05			
Weight Decay	0.01			
Warm-up Ratio	0.1			
Implement Layer	Q,K,V,Up,Down,Gate			
LoRA Rank r	64	128	8	12
LoRA Alpha α	128	256	16	24
Experts Number	-		8	
Top-K	-		2	
Balance Loss Factor	-		0.01	
Projection Rank d	-			32
Weight Balance Scale β	-			1

Table 8: Experimental Hyperparameter Configurations for Inter-Domain Learning with Various PEFT Methods. Experiments are conducted on four Nvidia A100 GPUs.

A.3.2 Intra-Domain Tasks Learning

Hyperparameters	LoRA/DoRA	AsyLoRA	MoLoRA	MALoRA
Optimizer	AdamW			
Learning Rate	4e-4			
Epochs	2			
Batch Size	2			
Dropout	0.05			
Weight Decay	0.01			
Warm-up Ratio	0.1			
Implement Layer	Q,K,V,Up,Down,Gate			
LoRA Rank r	64	128	8	12
LoRA Alpha α	128	256	16	24
Experts Number	-		8	
Top-K	-		2	
Balance Loss Factor	-		0.01	
Projection Rank d	-			32
Weight Balance Scale β	-			1.25

Table 9: Experimental Hyperparameter Configurations for Common-Sense Multi-Task Learning with Various PEFT Methods. Experiments are conducted on a single NVIDIA A100 GPU.

A.4 The Impact of Hyperparameter β

Defining $\mathcal{L}$ as the loss function of fine-tuning, $W \in \mathbb{R}^{m\times n}$ as a linear matrix in the backbone model where the MALoRA modules are attached to. We have the partial derivatives of loss $\mathcal{L}$ for matrix $S_A \in \mathbb{R}^{d\times n}$ and $P_t \in \mathbb{R}^{r\times d}$:

$$\Delta W_t = \overline{B}_t P_t S_A \tag{13}$$

$$\frac{\partial \mathcal{L}}{\partial S_A} = \frac{\partial W}{\partial S_A}\frac{\partial \mathcal{L}}{\partial W}, \quad \frac{\partial \mathcal{L}}{\partial P_t} = \frac{\partial \mathcal{L}}{\partial W}\frac{\partial W}{\partial P_t} \tag{14}$$

$$\frac{\partial \mathcal{L}}{\partial S_A} = \sum_{t=1}^{N} G_t(I^T \otimes (P_t\overline{B}_t))_{dn\times mn}\frac{\partial \mathcal{L}}{\partial W} \tag{15}$$

$$\frac{\partial \mathcal{L}}{\partial P_t} = G_t\frac{\partial \mathcal{L}}{\partial W}(\overline{B}_t^T \otimes S_A)_{mn\times rd} \tag{16}$$

Here, operator "$\otimes$" stands for the Kronecker Product. Since the norm of P_t is proportional to $1/\beta$, and the norm of S_A is proportional to β, so does the norm of gradient:

$$\|\frac{\partial \mathcal{L}}{\partial S_A}\| \propto \frac{1}{\beta}, \quad \|\frac{\partial \mathcal{L}}{\partial P_t}\| \propto \beta \tag{17}$$

Hence, an elevated value of β leads to an augmentation in the magnitude of $\|\nabla P_t\|$ and a concomitant reduction in the magnitude of $\|\nabla S_A\|$, thereby enhancing the model's capacity to discern task-specific discrimination. In contrast, a diminished value of β endows the model with a propensity to primarily capture commonalities across datasets.

A.5 Insufficient Learning Ability of AsyLoRA

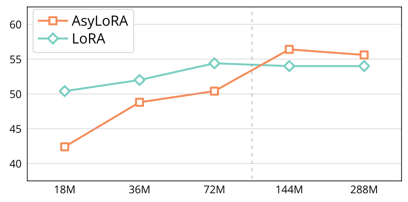


Figure 7: Performance Variations of LoRA and Asymmetry LoRA with Respect to Rank on the Math Reasoning Task MGSM (Shi et al., 2022).The ticks on the x-axis represent the number of trainable parameters in Asymmetry LoRA and LoRA.

In this section, we discuss the reasons for the poor performance of Mixture-of-Asymmetry-LoRA (MoAsyLoRA). In experts of MoAsyLoRA, the rank r is doubled while the matrix A remains frozen during fine-tuning. Consequently, the generalization boundary of these experts aligns with MoLoRA rather than expanding. To further investigate the mechanism behind performance degradation, we conduct an ablation study of the rank r for both Asymmetry LoRA and LoRA methods in math reasoning tasks, as shown in Figure 7. Under complex tasks like math reasoning, Asymmetry LoRA achieves better generalization only when the rank is sufficiently high. This can be attributed to the frozen down projection matrix, which randomly projects information dimensions in Asymmetry LoRA. When the model lacks the capability to fine-tune effectively, it compensates by increasing the dimension of the random projection, overriding the optimization of the down projection matrix in LoRA. However, experts in MoLoRA and MoAsyLoRA typically operate at low ranks, resulting in significant generalization drawbacks between Asymmetry LoRA and LoRA experts.

A.6 Training Latency Comparison

Method	Forward	Backward	Optimize	Total	
				Token(μs)	Step(s)
LoRA	4.246	9.229	0.188	833.9	13.663
DoRA	4.973	11.514	0.240	1020.8	16.726
MoLoRA	4.649	11.902	1.017	1072.3	17.568
MALoRA	4.288	9.523	0.876	896.4	14.687

Table 10: Training Latency Components of PEFT Methods in s/Step and μs/Token. A single step of training processes 16,384 input tokens. The single-step training latency of MALoRA accounts for only about 85% of that of MoLoRA, achieving a 1.2x speed improvement. Metrics are evaluated using four NVIDIA A100 GPUs.

A.6.1 Comparison of Computation Times

In this section, we compare the computation times of MALoRA and LoRA. MALoRA reduces theoretical computational complexity compared to LoRA under the same parameter budget in a constant sense. However, due to the architecture of MoE additional overhead from token distribution and CUDA optimizations, its practical latency is slightly higher than LoRA in practical application.

Suppose the model contains L layers. PEFT models are attached to K linear modules per layer, while each linear modules possess a input shape n and output shape m. Denote the rank of S_A as d, the rank of experts in MoLoRA as r, and the expanded LoRA rank in MALoRA as $\overline{r}$. MALoRA consists of N experts, with the top-K experts selected for inference. Given the same parameter budget, LoRA has the rank of $R = Nr$. The computational times of MALoRA compared to the original LoRA is as follows:

$$C_T(LoRA) = TLK(n+m)Nr \tag{18}$$

$$\begin{aligned} C_T(MALoRA) = TLK[&nN + nd+ \\ &K(d+m)\overline{r}] \end{aligned} \tag{19}$$

Since $d = \lambda Nr \ll m$, for simplicity, we approximate it to zero and assume $n = m$, while noticing $\overline{r}$ is determined by $\overline{r} = (2-\lambda)r$:

$$C_T(LoRA) \approx TLKn(2Nr) \tag{20}$$

$$\begin{aligned} C_T(MALoRA) \approx TLKn[&N(\lambda r + 1) \\ &+ K(2-\lambda)r] \end{aligned} \tag{21}$$

In our experimental setup, we set $K = 2$, $N = 8$, $r = 8$, and $\lambda = \frac{1}{2}$, leading to the following:

$$C_T(LoRA) = 128TLKn \tag{22}$$

$$C_T(MALoRA) = 64TLKn \tag{23}$$

Thus, the computational amount of LoRA is larger than that in MoLoRA. However, the MoE structure involves the distribution and collection of tensors among experts, whereas LoRA, because of its simplicity, may offer certain speed advantages in practical applications.

A.7 Rank Robustness

Table 11 shows the performance of MoLoRA and the proposed MALoRA under proportional ranks. As mentioned in Section 3.3, the value of λ is fixed at $\frac{1}{2}$, and the hyperparameter d is uniquely determined by λ, N and r. As the quantity of PEFT parameters increases, the dimension of S_A expands, which in turn diminishes the marginal benefit of fine-tuning S_A, potentially resulting in over-fitting. Consequently, it is advisable to select a larger β to emphasize the acquisition of distinctive representations in diverse tasks. From this perspective, by scaling β in proportion to the increase in rank r, we can effectively mitigate under- and over-fitting phenomena.

A.8 Hyperparameter Robustness

A.8.1 Hyperparameter Robustness in Intra-Domain Learning

Table 12 describes MALoRA's detailed performance in Figure 4(b). When choosing a reasonable asymmetric rank setting (e.g. $d = 32, 48$), in the vast majority of cases of β, MALoRA outperforms all baselines.

A.8.2 Universality under Different Task Combinations

We further validate the stability of MALoRA with respect to the hyperparameter β under more diverse sets of task combinations, as shown in Tables 13 and 14. In the FINANCE+CODE+ARC and

Method	r	d	β	MedMCQA	HumanEval	GSM8K	PhraseBank	ARC-C	ARC-E	E2E	AVG.
MoLoRA	4	-	-	42.4	23.8	34.0	69.3	58.8	77.4	66.1	53.12
	8			42.1	29.3	33.1	78.1	61.1	78.1	65.6	55.33
	16			43.4	32.9	33.4	79.4	61.5	78.0	66.0	56.19
	32			45.8	33.5	32.0	72.6	61.9	78.3	66.0	55.74
MALoRA	6	16	0.3	43.2	29.9	32.0	74.0	58.9	76.7	65.5	54.28
	12	32	1.0	42.3	32.9	34.1	79.8	60.0	78.5	66.3	56.27
	24	64	3.5	42.0	30.5	36.1	80.6	62.4	78.6	66.1	56.61
	48	128	5.0	43.9	34.8	35.2	76.7	62.2	78.1	65.9	56.69

Table 11: Performance of MoLoRA and Proposed MALoRA Under Proportional Ranks. In all configurations, MALoRA consistently outperformed the MoLoRA approach.

FINANCE+MED+BoolQ+ARC settings, with β values incrementing from 0.5 to 1.25. MALoRA outperforms the baselines of LoRA and MoLoRA across all test results. The experiment results indicate that the performance of various β demonstrates robustness, while selecting β with validation will leads to further improvement.

A.8.3 Guidelines for Selecting β

Carefully tuning the hyperparameter β will bring superior performance on stable gain of MALoRA. Although adjustment can be made with the help of the validation set, in this section, we conclude several empirical selection rules.

Relationship with the Amount of Trainable Parameters As the number of trainable parameters increases, a higher value of β is generally needed to achieve optimal performance, as shown in Table 11. When the shared subspace expands with more parameters, the risk of overfitting increases. To address this, a larger β emphasizes task-specific learning, effectively balancing the risks of under and overfitting.

Relationship with Task Relevance We conduct additional experiments to investigate the optimal β across varying levels of task similarity, shown in Table 15. It can be seen that as the diversity and the similarity of tasks increases, the value of β should be raised to improve the learning of P_t, thus better capturing the independent characteristics of different tasks within the shared subspace.

Relationship with λ Hyperparameter λ represents the account for dimension d of the shared low-rank space span(S_A). A lower value of λ, which corresponds to an increase in $\overline{r}$, weakens the ability of the shared space to capture the characteristics of tasks. Under such conditions, the emphasis on learning task-specific components becomes more critical, requiring a larger value of β. For example, as shown in Table 12, the optimal β for decreasing d are 1, 1.25, 1.25, and 1.25, respectively. However, since λ is fixed to $\frac{1}{2}$ according to section 3.3, in the vast majority of our experiments, we do not need to pay much attention to this situation.

A.9 Generality

A.9.1 Generality to Larger-Scale Models

To validate MALoRA's scalability to larger-scale models, additional fine-tuning results on LLaMA-2 13B are shown in Table 16. MALoRA outperforms MoLoRA by 1.29%, demonstrating strong universality for larger-scale models. Similarly to what is observed on LLaMA-2 7B, multi-task learning on LLaMA-2 13B also exhibits stability on the hyperparameter β. As Table 17 presents, although selecting through a wide range of β, the performance of MALoRA models exceeds all the baselines.

A.9.2 Generality under Different Task Combinations

In this section, we provide details of experiments on various combinations of tasks, which obtain a series of decreasing task similarities to validate the stability of MALoRA across different dataset groups. As shown in Table 18, 19 and 20. The results demonstrate that our method outperforms baseline approaches in task combinations with varying degrees of similarity, showcasing its robustness in diverse scenarios.

r	d	β	**PIQA**	**OBQA**	**BoolQ**	**ARC-C**	**ARC-E**	**AVG.**
14	16	0.5	81.99	89.8	73.15	66.21	82.83	78.80
		1.0	81.28	87.4	79.69	65.74	81.31	79.08
		1.25	80.79	88.6	80.34	65.67	83.12	79.70
		2.0	79.98	87.8	78.44	66.89	81.65	78.95
12	32	0.5	79.76	89.6	82.87	64.17	81.82	79.64
		1.0	80.96	89.0	81.38	66.55	82.24	80.03
		1.25	81.83	89.6	81.56	66.55	82.83	80.47
		2.0	81.66	88.4	78.01	65.74	82.37	79.24
10	48	0.5	82.26	89.6	77.77	65.81	81.94	79.48
		1.0	82.32	89.0	76.54	68.12	82.62	79.72
		1.25	81.34	90.0	81.83	68.05	82.74	80.79
		2.0	81.56	88.4	82.17	67.78	82.58	80.50
8	64	0.5	79.11	88.8	81.01	64.17	79.04	78.43
		1.0	79.87	87.6	83.33	63.90	80.77	79.09
		1.25	81.23	87.8	77.98	65.40	82.15	78.91
		2.0	80.14	87.8	81.83	64.79	80.30	78.97

Table 12: Details of Hyperparameter Robustness Experiment of MALoRA Evaluated on Intra-Domain Tasks.

FINANCE+CODE+ARC				
β	**0.5**	**0.75**	**1.0**	**1.25**
LoRA	60.69			
MoLoRA	61.09			
MALoRA	**61.91**	**62.16**	**62.00**	**62.12**

Table 13: Results of FINANCE+CODE+ARC Task Combination with Tuned β. Evaluations are conducted on PhraseBank, HumanEval, ARC-C and ARC-E benchmarks. This group of tasks shares moderate relevance, with mutual promotion observed between Finance and Code learning.

FINANCE+MED+BoolQ+ARC				
β	**0.5**	**0.75**	**1.0**	**1.25**
LoRA	60.69			
MoLoRA	61.09			
MALoRA	**61.91**	**62.16**	**62.00**	**62.12**

Table 14: Results of FINANCE+MED+BoolQ+ARC Task Combination with Tuned β. Evaluations are conducted on PhraseBank, MedMCQA, BoolQ and OBQA benchmarks. Compared to previous tasks, this group of tasks is more diverse and has lower relevance.

Tasks	**LoRA**	**MoLoRA**	**MALoRA**	**Possible** β
ARC + OBQA	78.18	79.01	**79.46**	0.5
Finance + Code + ARC	60.69	61.09	**61.91**	0.5
Finance + Medicine + BoolQ + OBQA	68.95	69.06	**69.85**	0.75
Inter-Domain	54.72	55.33	**56.27**	1.0

Table 15: Possible β Choices for Different Task Combinations. As the similarity of tasks decreases, the optimal choice for β increases.

Method	**PIQA**	**OBQA**	**BoolQ**	**ARC-C**	**ARC-E**	**AVG.**
LLaMA-2 13B	62.40	65.60	62.23	50.78	66.84	61.57
LoRA	84.39	89.00	80.40	69.61	85.15	81.91
AsyLoRA	**84.77**	88.40	80.28	69.75	86.11	81.86
MoLoRA	**84.77**	89.60	76.06	70.63	**86.91**	81.59
MALoRA	84.39	**89.80**	**81.99**	**71.86**	86.36	**82.88**

Table 16: Results of Intra-Domain Learning, using LLaMA-2 13B as the Backbone Model

β	**0.5**	**0.75**	**1.0**	**1.25**
MALoRA	82.32	82.58	82.44	82.88

Table 17: Results of Intra-Domain Learning, using LLaMA-2 13B as the Backbone Model with a serise value of β

OBQA+ARC				
Method	**OBQA**	**ARC-C**	**ARC-E**	**AVG.**
LoRA	88.2	64.65	81.69	78.18
MoLoRA	88.6	**66.69**	81.74	79.01
MALoRA	**89.8**	66.35	**82.24**	**79.46**

Table 18: Results of Different Methods in OBQA+ARC Task Combination.

PhraseBank+HumanEval+ARC					
Method	**PhraseBank**	**HumanEval**	**ARC-C**	**ARC-E**	**AVG.**
LoRA	73.44	**28.05**	**61.79**	79.46	60.69
MoLoRA	76.95	26.22	61.25	79.92	61.09
MALoRA	**79.47**	26.83	61.52	**79.84**	**61.91**

Table 19: Results of Different Methods in FINANCE+CODE+ARC Task Combination. The complexity of the tasks in this group has increased, and there is a phenomenon of mutual promotion between the learning of FINANCE and CODE.

FINANCE+MEDICINE+BoolQ+OBQA					
Method	**PhraseBank**	**MedMCQA**	**BoolQ**	**OBQA**	**AVG.**
LoRA	68.00	44.04	76.36	87.4	68.95
MoLoRA	69.10	43.29	**77.65**	86.2	69.06
MALoRA	**70.08**	**44.50**	75.81	**89.0**	**69.85**

Table 20: Results of Different Methods in FINANCE+MEDICINE+BoolQ+OBQA Task Combination. This group of tasks are more diverse and has lower relevance.