RotMoLE: Enhancing Mixture of Low-Rank Experts through Rotational Gating Mechanism

Mengyang Sun[1*] **Maochuan Dou**[2] **Tao Feng**[1] **Dan Zhang**[3]
Yihao Wang[2] **Junpeng Liu**[4] **Yifan Zhu**[5†] **Jie Tang**[1†]

[1]Tsinghua University [2]Beijing Information Science and Technology University
[3]National University of Singapore [4]Hong Kong University of Science and Technology (Guangzhou)
[5]Beijing University of Posts and Telecommunications
sunmy19@mails.tsinghua.edu.cn {2023011446, 2022011219}@bistu.edu.cn
fengtao.hi@gmail.com zhangdan25@nus.edu.sg junpengliu@hkust-gz.edu.cn
yifan_zhu@bupt.edu.cn jietang@tsinghua.edu.cn

Abstract

While Large Language Models (LLMs) are commonly fine-tuned to handle domain-specific tasks before being applied to vertical applications, adapting them to complex scenarios with diverse specialized knowledge remains challenging. Meanwhile, Mixture-of-Experts (MoE) architecture has risen as a crucial paradigm for training LLMs, and some recent works have also incorporated MoE into Parameter-Efficient Fine-Tuning (PEFT) to propose the Mixture of Low-rank Experts (MoE-LoRA), to enhance the power of low-rank adapters for learning complicated knowledge. However, conventional gating mechanisms in MoE typically apply only a scalar reweighing to selected experts, thereby limiting their underlying capacity of representation and generalization. Motivated and enabled by the low-rank structures in MoE-LoRA, we propose RotMoLE, a specialized MoE framework for low-rank experts featuring an additional rotation gate. Beyond simple scaling, RotMoLE implements a rotation mechanism for each selected expert, enabling superior expert exploitation and specialization for learning diverse data, especially when expert candidates are limited. Results on complex training scenarios validate our effectiveness. Code is available at https://github.com/smyconan/RotMoLE.

1 Introduction

Through Mixture-of-Experts (MoE) [Jacobs et al., 1991] architecture and Parameter-Efficient Fine-Tuning (PEFT) [Zhang et al., 2025a] technique, large language models (LLMs) exhibit their capabilities of accommodating massive general knowledge and rapidly adapting to specific vertical domains. As noted by Cai et al. [2025] and Zhang et al. [2025b], MoE serves as a sparse framework for efficiently scaling LLMs. Through top-k selection and weighted aggregation of experts, MoE exhibits its significant capacity and flexibility across diverse fields, especially for complex and multi-task scenarios. Specifically, top-k expert selection in MoE enables sparse activation, which thereby not only boosts the computational efficiency [Shazeer et al., 2017], but also facilitates the decoupling and isolation of specialized domain knowledge [Ren et al., 2023]. Additionally, the weighted expert aggregation acts as a magnitude scaling process applied to the outputs of selected experts based on their respective gate values, thereby further expanding the representational capacity of MoE [Nie et al., 2021]. The whole paradigm is formulated as $y = \sum_{i \in TopK(x)} g_i E_i(x)$, where g_is denote the gate values and we typically normalize $\sum_{i \in TopK(x)} g_i = 1$ to ensure numerical stability.

[*] The work was done while the author interned at Zhipu.AI.
[†] These authors are corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

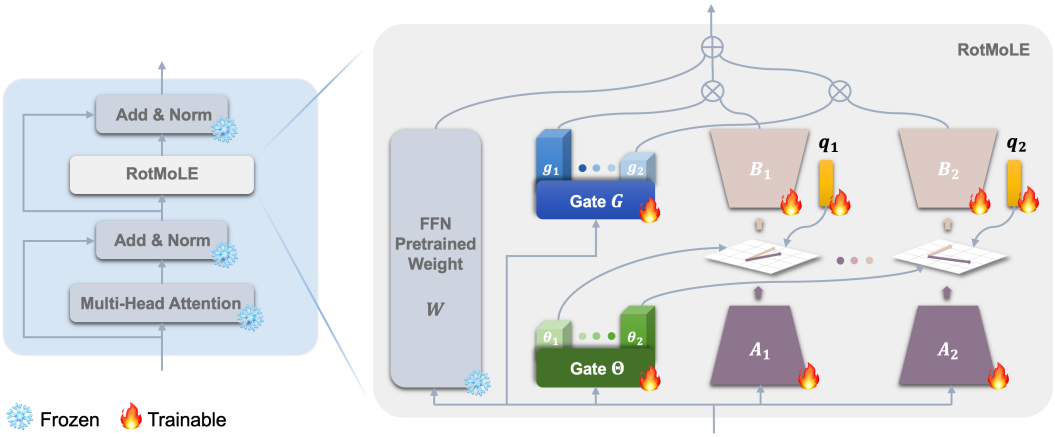


Figure 1: The structure of our Rotatable Mixture of Low-rank Experts (i.e., RotMoLE): To model a rotation operation for each expert beyond the simple gate-value-based scaling, we implement a rotation gate $\theta = \Theta(x)$ as well as a set of learnable vectors $\boldsymbol{q}$, for generating rotation angles and rotation planes for each low-rank expert. Specifically, for the LoRA expert's rank equal to r, we conduct a 2D rotation for the r-dimensional embeddings in a 2D sub-plane, given by $W = W_0 + \sum_{i \in TopK(x)} g_i B_i R_i A_i$, where R_i denotes the r-dimensional rotation matrix constructed by θ_i and $\boldsymbol{q}_i$.

Furthermore, as the Low-rank Adaptation (LoRA) [Hu et al., 2022] gains increasing traction in the field of PEFT, recent researchers have also incorporated MoE into LoRA and proposed Mixture of Low-rank Experts (MoE-LoRA) [Dou et al., 2024, Li et al., 2024]. Specifically, conventional LoRA introduces a trainable pair of low-rank matrices $A_{r\times d}$ and $B_{d\times r}$ ($r \ll d$) to a fixed pretrained module $W_{d\times d}$, given by $W = W_0 + BA$; while MoE-LoRA similarly introduces a series of LoRA modules B_i and A_i, as well as a routing module $g = G(x)$ to W, given by $W = W_0 + \sum_{i \in TopK(x)} g_i B_i A_i$. The incorporation of multiple low-rank modules in MoE-LoRA demonstrably improved the performance of PEFT in various scenarios, such as multi-task learning [Liu et al., 2024b] and continual learning [Ge et al., 2025], etc. However, there still remain challenges in modeling diverse specialized knowledge in complicated downstream scenarios using such parameter-efficient methods like MoE-LoRA.

Most MoE frameworks employ a simple scaling gate to select and weigh experts, which effectively modulates the magnitude of experts' respective contributions. However, it fundamentally lacks the capacity to perform complex spatial transformations on those selected experts. Consequently, it is insufficient for modeling intricate exploitation strategies of candidate experts required by diverse and complicated scenarios, such as multilingual or multi-task learning, especially when candidate experts are limited. Specifically, scalar gate values are only able to scale the norms of expert outputs along their fixed spatial directions. This limitation restricts the solution space formed by the weighted aggregation of selected experts. As a result, those routers are only limited to determining *how much* an expert contributes, while overlooking *how diverse* it can contribute.

A natural solution is to introduce an enhanced gate module capable of providing each selected expert with a more powerful transformation function beyond scaling. However, this brings a substantial demand for additional trainable parameters. For example, an unconstrained linear transformation of a d-dimensional vector requires a $d \times d$ matrix. Therefore, with n d-dimensional experts, this would rely on d^3n trainable gating parameters, which are disproportionately large for a routing module, and may even be far larger than expert modules themselves. Dynamically constructing these matrices from fewer coefficients during inference may alleviate the storage burden, but it still leads to a temporary $d \times d$ memory usage and a computational cost of d-dimensional matrix construction and multiplication. Given that the hidden dimension d in current mainstream LLMs typically exceeds $4k$ [Zhao et al., 2023], such an approach is effectively prohibitive compared to its benefits. Notably, the low-rank structure of LoRA offers an opportunity to potentially implement complex expert transformations in MoE-LoRA that go beyond simple scaling: Since $r \ll d$, an r-dimensional linear transformation function requires only an $r \times r$ matrix if we apply it after module A, such as $W = W_0 + \sum_{i \in TopK(x)} B_i G_i A_i$, where G_i is an $r \times r$ matrix generated by a gate module with only dnr^2 trainable parameters. However, compared to the low-rank expert modules themselves which totally consist of $2ndr$ parameters, it is still a disproportionately large gate.

Motivated by the observation that any transformation between two specific non-zero vectors can be formulated as a 2D transformation within the sub-plane spanned by the two vectors, and thus can be further decomposed into a pure scaling operation and a pure rotation within the plane, we propose Rotatable Mixture of Low-rank Experts (i.e., RotMoLE), a specialized MoE framework for low-rank experts featuring an additional rotation gate. Specifically, RotMoLE assigns not only a scaling value g_i, but also a rotation coefficient θ_i to each candidate expert. For the case where the expert rank r equals 2, a 2D rotation matrix can be naturally derived as:

$$R_i = \begin{pmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{pmatrix};$$

While for r larger than 2, the rotation operation is much more complex to formulate since it requires more variables beyond a single θ_i to specify the rotation sub-plane $\mathcal{P}_i$ for expert i. Therefore, we further employ a small set of rn trainable parameters to determine a temporary rotation plane $\mathcal{P}_i$ for each expert and thereby construct the $r \times r$ rotation matrix R_i based on $\mathcal{P}_i$ and θ_i. As a result, our proposed RotMoLE can be expressed as $W = W_0 + \sum_{i \in TopK(x)} g_i B_i R_i A_i$, while the total number of extra parameters we introduce is only dn for LoRA ranks equal to 2, and $(d + r)n$ for LoRA ranks larger than 2, respectively. The main structure of RotMoLE is exhibited by Figure 1.

The main concepts of this paper are related to the fields of MoE in LLMs, PEFT, LoRA, and MoE-LoRA, etc. To the best of our knowledge, few existing works in those fields have considered the expressiveness limitation of MoE-LoRA gates and integrated rotational gating. A closely related work is RadarGate Guo et al. [2025a], which introduces rotation into MoE-LoRA but directly rotates the outputs of LoRA experts rather than exploiting their low-rank structure to simplify the computation. Please refer to Appendix A for related works of those fields.

Our main contributions are summarized as follows:

- We emphasize the limitation of existing MoE routers, which only perform a norm scaling for each selected expert, lacking the capability to perform complex transformations on experts for diverse scenarios, especially when experts are limited.
- We indicate the potential of implementing more complex transformations for routing low-rank experts in MoE-LoRA, and propose RotMoLE as a specialized MoE-LoRA framework featuring an additional rotation gate.
- We conduct a series of multi-task and multilingual experiments under various circumstances, illustrating the effectiveness and generalization of our RotMoLE over conventional MoE-LoRA baselines.

2 Methodology

Our RotMoLE serves as a specialized framework of MoE-LoRA, relying on decomposing an unconstrained transformation of candidate experts into a pure scaling operation and a pure rotation within a 2-dimensional plane space. Specifically, we treat the conventional gate modules in MoE as the scaling operators, so the weighted aggregation based on their generated gate values can be regarded as a normalization to preserve the scale consistency of hidden states during layer-up computation; To model rotation, we introduce an extra gate module named rotation gate, which generates rotation angle values θs for selected experts. A specific low-rank rotation matrix R_i is subsequently derived from the respective angle value θ_i for each expert i, and then multiplying the expert in the middle of two low-rank matrices A_i and B_i, given by $W = W_0 + \sum_{i \in TopK(x)} g_i B_i R_i A_i$. We claim that performing customized rotation operations for each expert can further expand the global solution

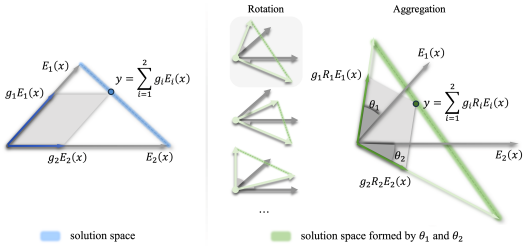


Figure 2: Illustrations of the MoE solution space through a case study in a 2-dimensional space with two candidate experts.

space of MoE. We illustrate this through a simple case study in Figure 2. The left part of Figure 2 shows the solution space of a conventional 2-expert MoE structure: Given the outputs of candidate experts $E_1(x)$ and $E_2(x)$, and their respective gate values g_1 and g_2 ($g_1 + g_2 = 1$), the weighted-sum solution space locates only within the blue line; While the right part of Figure 2 demonstrates the case of integrating a rotation gate (which is our proposed method): Given respective rotation angles θ_1 and θ_2, outputs of candidate experts are rotated by θ_i first, and then weighted summed by their respective gate values. The solution space of weighted-summing thus transfers to the green line. As a result, through dynamic and customized θs, the solution space of MoE is expanded.

We elaborate on our RotMoLE in the following four aspects: (1) the theoretical foundation of RotMoLE; (2) the rotation gate module generating angle values; (3) the algorithm of conducting low-rank rotation through rotation matrix, typically for expert ranks equal to 2; and (4) the solution we propose for rotation matrix construction for larger expert ranks.

2.1 Theoretical Foundation

Our RotMoLE is based on our observation that any transformation between two specific non-zero vectors, no matter originally linear or non-linear, can be reversely written as a 2D transformation within the sub-plane spanned by the two vectors, and thus can be further decomposed into a pure scaling operation and a pure rotation within the plane. To specify this, we first denote the two random non-zero d-dimensional vectors as $\boldsymbol{u}$ and $\boldsymbol{v}$, and two orthogonal basis vectors of the plane $\mathcal{P}$ spanned by $\boldsymbol{u}$ and $\boldsymbol{v}$ as $\boldsymbol{e_1}$ and $\boldsymbol{e_2}$, where $|\boldsymbol{e_1}| = |\boldsymbol{e_2}| = 1$ and $\boldsymbol{e_1} \cdot \boldsymbol{e_2} = 0$. Then we can rewrite $\boldsymbol{u}$ and $\boldsymbol{v}$ as:

$$\begin{aligned}\boldsymbol{u} &= (\boldsymbol{u} \cdot \boldsymbol{e_1})\boldsymbol{e_1} + (\boldsymbol{u} \cdot \boldsymbol{e_2})\boldsymbol{e_2} = u_1\boldsymbol{e_1} + u_2\boldsymbol{e_2};\\ \boldsymbol{v} &= (\boldsymbol{v} \cdot \boldsymbol{e_1})\boldsymbol{e_1} + (\boldsymbol{v} \cdot \boldsymbol{e_2})\boldsymbol{e_2} = v_1\boldsymbol{e_1} + v_2\boldsymbol{e_2},\end{aligned}$$

and so that we can define $\boldsymbol{u}_\mathcal{P} = [u_1, u_2]^T$ and $\boldsymbol{v}_\mathcal{P} = [v_1, v_2]^T$ as the views of $\boldsymbol{u}$ and $\boldsymbol{v}$ under the condition of plane $\mathcal{P}$. Afterwards, these two vectors can be further expressed by:

$$\boldsymbol{u}_\mathcal{P} = \sqrt{u_1^2 + u_2^2} \cdot [\cos\theta_u, \sin\theta_u]^T; \boldsymbol{v}_\mathcal{P} = \sqrt{v_1^2 + v_2^2} \cdot [\cos\theta_v, \sin\theta_v]^T,$$

where $\theta_u = \arccos \frac{u_1}{\sqrt{u_1^2+u_2^2}}$ and $\theta_v = \arccos \frac{v_1}{\sqrt{v_1^2+v_2^2}}$. Finally, we can formulate the transformation from $\boldsymbol{u}_\mathcal{P}$ to $\boldsymbol{v}_\mathcal{P}$ as:

$$\boldsymbol{v}_\mathcal{P} = \frac{\sqrt{v_1^2 + v_2^2}}{\sqrt{u_1^2 + u_2^2}} \cdot Rotate(\boldsymbol{u}_\mathcal{P}, \theta_v - \theta_u) = \frac{\sqrt{v_1^2 + v_2^2}}{\sqrt{u_1^2 + u_2^2}} \cdot \begin{pmatrix}\cos(\theta_v - \theta_u) & -\sin(\theta_v - \theta_u)\\ \sin(\theta_v - \theta_u) & \cos(\theta_v - \theta_u)\end{pmatrix} \cdot \boldsymbol{u}_\mathcal{P},$$

indicating that we first rotate the vector $\boldsymbol{u}_\mathcal{P}$ by the angle of $\theta_v - \theta_u$ in plane $\mathcal{P}$, and then scale it by a scalar $\frac{\sqrt{v_1^2+v_2^2}}{\sqrt{u_1^2+u_2^2}}$.

2.2 Rotation Gate

Similar as the scaling gate $g = G(x) = Softmax(x \cdot W_g)$, we also implement a linear module for the rotation gate. Here we denote it as $x \cdot W_\theta$. Instead of employing a $Softmax$ function which targets for a sum-to-1 normalization among all n experts, our rotation operation naturally does not require a sum-to-1 normalization since the operation of rotating each expert is respectively independent. However, we notice that the rotation angles exhibit periodicity. This non-monotonic property may significantly affect model training. Therefore, we implement an independent normalization for each angle to restrict it within a single period. The whole rotation gate is expressed as follows:

$$\theta = \Theta(x) = 2\pi \cdot Sigmoid(x \cdot W_\theta) - \pi,$$

where the $Sigmoid$ function normalizes per-expert value to $[0, 1]$, and then it is mapped to $[-\pi, \pi]$. We initialize the trainable parameters W_θ to 0 since we regard the default condition as the condition without any rotation. Specifically, $Sigmoid$ function is more sensitive to input variations within the

nearby range of 0, while more stable to larger positive or negative inputs. As a result, θ is sensitive to trainable parameters W_θ around $2\pi \cdot Sigmoid(0) - \pi = 0$, while more stable around $-\pi$ or π when W_θ is relatively large. This will theoretically encourage both early-stage convergence and late-stage stability of our rotation gate, boosting the training performance of RotMoLE.

For sparse expert activation, instead of conducting top-k expert selection based on rotation angle values, we still follow the convention of previous MoE frameworks to perform sparse activation only based on scaling gate values g_is. In other words, k experts with the largest g_i values are activated. This is due to the concept that only scaling gate values g_is indicate the relative importance of each expert, while the rotation gate only contributes to the diversity of expert exploitation and generalization.

2.3 2-Dimensional Rotation

Two essential components for conducting a rotation operation in a d-dimensional space are the rotation angle and the rotation plane (i.e., a 2-dimensional subspace). For the case where expert rank equals 2, the entire space is 2-dimensional, so the rotation angle serves as the only coefficient for modeling a rotation operation. A 2-dimensional rotation can be expressed by a 2-dimensional matrix R_i, and therefore, our proposed RotMoLE can be derived as the follows:

$$W = W_0 + \sum_{i \in TopK(x)} g_i B_i R_i A_i = W_0 + \sum_{i \in TopK(x)} g_i B_i \begin{pmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{pmatrix} A_i,$$

where $g = G(x)$ and $\theta = \Theta(x)$, indicating the scaling and the rotation gate modules, respectively.

2.4 Rotation for Larger LoRA Ranks

To implement rotation operation for expert rank r larger than 2, we follow the theoretical foundation in Section 2.1, which indicates that the r-dimensional rotation operation inside each r-rank LoRA expert can be formulated as a 2-dimensional rotation by constructing a rotation plane $\mathcal{P}_i(x)$ for each expert i and each query x, spanned by the input embedding $\boldsymbol{u}$ (i.e., $A_i x$, since R_i is in the middle of A_i and B_i) and output embedding $\boldsymbol{v}$. However, as the rotation output $\boldsymbol{v}$ is unknown and is just what we aim to obtain, this rotation plane can not be determined only by $\boldsymbol{u}$. As a result, we also need to model rotation planes as variables, just as the way we model rotation angles.

Specifically, as in expert i the input embedding $\boldsymbol{u}$ for rotation (i.e., $A_i x$) definitely lies in its rotation plane, we consider one of the plane basis $\boldsymbol{e_1}$ as $\frac{A_i x}{|A_i x|}$, leaving the other r-dimensional basis $\boldsymbol{e_2}$ as the only variable. Then, instead of applying another input-related gate to generate a specific $\boldsymbol{e_2}$ for each expert through a linear mapping $x_{d\times 1} \rightarrow n\boldsymbol{e_2}_{r\times 1}$ which requires dnr extra trainable parameters, we implement it in a lighter way. Specifically, Luo et al. [2024] claims that since each expert is designed for specializing in a specific domain of knowledge, input embeddings routed to the same expert are expected to be more similar than those routed to distinct experts, and so are their output embeddings. Based on that claim, the inputs xs routed to the same LoRA expert i are similar and can be regarded as a relatively aggregated cluster, and so are the $A_i x$ embeddings outputted by matrix A_i. As a result, the dispersion level of this $A_i x$s cluster indicates the diversity within expert i, and finally their respective rotation operations can therefore be light-weighted as functions enhancing this diversity to achieve better expert exploitation. To model this diversity enhancement, we constrain and simplify our rotation operation in expert i as a rotation only towards or away from the center of expert i's cluster, which also means that we require the rotation plane $\mathcal{P}_i(x)$ must pass through this cluster center, leveraging the center embedding as the other essential support for determining $\mathcal{P}_i(x)$ besides $A_i x$. Specifically, we employ a learnable r-dimensional vector $\boldsymbol{q_i}$ for each expert i as its cluster center, and construct the specific rotation plane $\mathcal{P}_i(x)$ for each embedding $A_i x$ using both $\boldsymbol{q_i}$ and $A_i x$, given by $\mathcal{P}_i(x) = span(\boldsymbol{q_i}, A_i x)$. Considering that $\boldsymbol{q_i}$ and $A_i x$ may not be orthogonal, we treat $\frac{A_i x}{|A_i x|}$ as $\boldsymbol{e_1}$ and the component of $\boldsymbol{q_i}$ orthogonal to $A_i x$ as $\boldsymbol{e_2}$. The whole procedure of constructing the r-dimensional rotation matrix R_i can be illustrated by the following equations:

$$\boldsymbol{e_1}_i = \frac{A_i x}{|A_i x|};$$

$$\boldsymbol{e_2}_i = \frac{\boldsymbol{e_2}_i{}^*}{|\boldsymbol{e_2}_i{}^*|}, \text{ where } \boldsymbol{e_2}_i{}^* = \boldsymbol{q_i} - (\boldsymbol{q_i} \cdot \boldsymbol{e_1}_i)\boldsymbol{e_1}_i;$$

$$R_i = (\boldsymbol{e_1}_i \quad \boldsymbol{e_2}_i) \cdot \begin{pmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{e_1}_i^T \\ \boldsymbol{e_2}_i^T \end{pmatrix},$$

where $g = G(x)$ and $\theta = \Theta(x)$. Here W_g, W_θ and $\boldsymbol{q}$ are learnable parameters, totally $2dn + rn$. Basically, these equations convert a 2-dimensional rotation matrix into its representation within the r-dimensional space via two planar basis vectors, and subsequently apply it to the r-dimensional inputs.

2.5 Theoretical Limitations

We acknowledge the theoretical limitation of RotMoLE, which is the **restricted transformation freedom and known condition limitation**. Specifically, as illustrated in Section 2.1, we rewrite any d-dimensional transformation into a pure scaling operation and a pure rotation within a 2D plane. Therefore, the freedom of an arbitrary d-dimensional transformation theoretically depends on the freedom of deciding the scalar, the rotation angle, and the rotation plane during inference. However, in our design the scaling factor g and rotation angle θ are determined by a linear module and a simple activation function, while the rotation plane $\mathcal{P}$ is determined by method in Section 2.4, where only the vector $\boldsymbol{u}$ is known with the unknown $\boldsymbol{v}$ substituted by an estimated vector $\boldsymbol{e_2}$. All these procedures restrict the range of values of g, θ, and $\mathcal{P}$, leading to a restriction of original d-dimensional transformation freedom. Although complexifying the related modules that generate g, θ, and $\mathcal{P}$ can expand their range of values to further approach free deciding, it is a trade-off in practice due to the efficiency.

3 Experiments

We conduct a series of comparative experiments to evaluate the effectiveness of RotMoLE across various complex scenarios, including multilingual and multi-task learning. To ensure fine-tuning complexity, we adopt a joint learning manner for all experiments, which means we jointly train on the mixture of all languages or tasks, named as Multilingual or Multi-task Joint Learning (MJL). We compare our RotMoLE with the conventional MoE-LoRA baseline which implements only a scaling gate $g = G(x)$, as well as some recent MoE-LoRA variants with specific structural designs.

Considering that RotMoLE involves more trainable parameters than the baseline, such as the rotation gate $\theta = \Theta(x)$ or the center vectors $\boldsymbol{q}$, we also perform a comparative experiment for ablation study between our RotMoLE and a size-equivalent MoE-LoRA with an MLP gate, to further illustrate the pure effectiveness of our structural design; Another ablation study we conduct is to verify RotMoLE under different model sizes. The two ablation studies are presented in Appendix E.

3.1 Overall Experimental Setup

Our proposed method is specifically designed for two circumstances: (1) LoRA expert ranks equal to 2, where we implement only a rotation gate; and (2) LoRA expert ranks larger than 2, where we implement a rotation gate along with n r-dimensional trainable vectors. Accordingly, our experiments cover both settings. Besides, we also conduct experiments under both dense MoE and sparse MoE configurations, demonstrating whether our proposed RotMoLE is influenced by the top-k sparse activation. Here we denote the total number of LoRA experts, the number of top experts selected during each forwarding, and the per-expert rank as n, k, r respectively. To simulate a resource-constrained scenario where experts are shared among multiple tasks, we explicitly constrain that $\frac{n}{k} < N_{task}$, where N_{task} denotes the number of tasks or languages we jointly train. We claim that our proposed RotMoLE provides extra diversity to those task-shared experts by further distinguishing different assigned tasks through rotation operations, and therefore can enhance the overall performance of MJL. For more details of experimental setup, please refer to Appendix B.

3.2 Multi-task Joint Learning

3.2.1 Mixture of QA Tasks

We mix three Question Answering tasks from various fields to constitute a mixture of Multiple QA tasks for our experiments, including CommonsenseQA [Talmor et al., 2019] (CSQA), Open-

BookQA [Mihaylov et al., 2018] (OBQA), and SocialIQA [Sap et al., 2019] (SIQA). Please refer to Appendix C for their detailed introduction. To cover the two circumstances that LoRA ranks both equal to and larger than 2, we respectively set rank r to 2 and 3 in our experiments. We also implement both dense MoE and sparse MoE settings, which consist of 2 experts with top-2 experts activated, and 4 experts with top-2 experts activated, respectively. We train the mixture on Llama-3.2-3B for around 4,000 steps, and on Qwen-2.5-3B for around 5,500 steps, ensuring they are fully updated. The batch size is 135, meaning we train 45 samples for each QA task per step.

We present our overall performances in Table 1, along with detailed per-task performances and the relative standard deviations (Std./Avg.) across tasks. The evaluation metric for QA is accuracy. It is observed that: (1) Our RotMoLE exhibits consistent superiority over the MoE-LoRA baseline (MoE-LoRA) across almost all configurations. Specifically, out of 8 experimental settings, RotMoLE achieves both higher average and per-task accuracy in 7 cases, with only one exception in configuration $\{r = 3, n = 4, k = 2\}$ under Llama-3.2-3B for OpenBookQA task; (2) Quantitatively, RotMoLE can roughly enhance the jointly learning performance on mixed QA tasks by about $1\% \sim 10\%$, averagely 5%; (3) The task-balancing performance of RotMoLE is overall equivalent or slightly better than that of the baseline, according to their relative standard deviation values across tasks; and (4) In most cases except $r = 2$ under Qwen-2.5-3B, the performance improvement of RotMoLE over the baseline is more pronounced when the number of experts is very limited, such as $n = 2$. This phenomenon indicates the effectiveness of our rotation gate mechanism in boosting per-expert diversity of MoE, since a limited number of experts is more likely to lead to task-shared experts which benefit more from our diversity-boosting design, as we may distinguish tasks by rotation within the same expert. Appendix D also presents the convergence performances of this experiment.

Table 1: Performance of Jointly Fine-Tuning on Mixed QA Tasks

Base Model	r	n, k	Method	CSQA	OBQA	SIQA	Avg.	Boost ↑	Std./Avg.
Llama-3.2-3B	2	2, 2	MoE-LoRA	34.73	42.60	42.48	39.93	6.74%	0.09
			RotMoLE	**40.21**	**43.80**	**43.86**	**42.62**		**0.04**
		4, 2	MoE-LoRA	36.94	40.80	41.15	39.63	4.82%	0.05
			RotMoLE	**38.90**	**41.80**	**43.91**	**41.54**		0.05
	3	2, 2	MoE-LoRA	38.57	37.00	40.43	38.67	10.03%	0.04
			RotMoLE	**43.57**	**41.80**	**42.27**	**42.55**		**0.02**
		4, 2	MoE-LoRA	35.95	**41.20**	36.34	37.83	4.10%	0.06
			RotMoLE	**37.67**	38.20	**42.27**	**39.38**		**0.05**
Qwen-2.5-3B	2	2, 2	MoE-LoRA	61.18	56.40	56.09	57.89	4.61%	**0.04**
			RotMoLE	**64.13**	**56.80**	**60.75**	**60.56**		0.05
		4, 2	MoE-LoRA	60.20	52.60	55.48	56.09	10.13%	0.06
			RotMoLE	**64.05**	**60.40**	**60.85**	**61.77**		**0.03**
	3	2, 2	MoE-LoRA	59.62	56.20	57.22	57.68	2.86%	**0.02**
			RotMoLE	**62.24**	**57.60**	**58.14**	**59.33**		0.03
		4, 2	MoE-LoRA	63.06	57.00	56.65	58.91	1.34%	0.05
			RotMoLE	**63.88**	**57.60**	**57.63**	**59.70**		0.05

3.2.2 Mixture of GLUE Tasks

We also mix general NLU tasks from the GLUE benchmark [Wang et al., 2018] for MJL evaluation. To simplify our experiments, we select only three out of the nine tasks in GLUE: SST-2, MRPC, and QNLI. Details of those tasks are described in Appendix C. Here, we directly set up a sparse MoE configuration and a LoRA rank larger than 2, as they are more general-purpose and are more commonly used. Specifically, we set the rank r of each LoRA expert to 4, and implement two sparse configurations, $\{n = 3, k = 2\}$ and $\{n = 5, k = 3\}$. Both the configurations still follow $\frac{n}{k} < N_{task}$ to model a resource-constrained scenario. We train the mixture of the three tasks on Llama-3.2-3B for around 5,000 steps. The batch size is 30, namely 10 per task. Table 2 illustrates the overall and per-task results. Our proposed RotMoLE still exhibits a visible outperformance over MoE-LoRA baseline.

Table 2: Performance of Jointly Fine-Tuning on Mixed GLUE Tasks (Llama-3.2-3B, $r = 4$).

n, k	Method	SST-2	MRPC	QNLI	Avg.
3, 2	MoE-LoRA	89.33	**71.77**	66.52	75.87
	RotMoLE	**90.37**	71.07	**66.67**	**76.03**
5, 3	MoE-LoRA	88.76	69.19	61.63	73.19
	RotMoLE	**89.22**	**69.21**	**61.83**	**73.42**

3.3 Multilingual Joint Learning

3.3.1 Multilingual Title Generation

For multilingual joint learning, we adopt the MTG (Multilingual Text Generation) benchmark proposed by Chen et al. [2022], and take its multilingual title generation sub-task as one of our evaluations. Five languages – English, Chinese, Spanish, French, and German – are involved in this task (see Appendix C for details). We test three MoE configurations: $\{r = 3, n = 2, k = 2\}$, $\{r = 2, n = 4, k = 2\}$, and $\{r = 4, n = 5, k = 3\}$. We train the language mixture on Llama-3.2-3B for 16,000 steps using a batch size of 50 (namely 10 per language). Results are presented in Table 3. Here we measure the average performances by Rouge-1, Rouge-2, and Rouge-L scores [Lin, 2004]. To reveal per-language efficacy, we also exhibit Rouge-L scores for each language. In general, RotMoLE still achieves better performances than the MoE-LoRA baseline in terms of both average and per-language Rouge metrics in most cases. We also compare their convergence performances under one of the MoE configurations: $\{r = 4, n = 5, k = 3\}$ in Figure 3. It shows that both methods initially update equivalently during the first 8,000 steps, but the MoE-LoRA baseline exhibits an earlier convergence, and overfits much faster than our proposed RotMoLE. This further indicates that our rotation design may introduce some extra robustness to model against routers overfitting.

Table 3: Performance of Jointly Fine-Tuning on Mixed Multilingual Title Generation (Llama-3.2-3B).

r, n, k	Method	EN	ZH	ES	FR	DE	Avg.		
		Rouge-L					Rouge-1	Rouge-2	Rouge-L
3, 2, 2	MoE-LoRA	32.33	23.87	27.80	25.28	22.79	29.63	11.76	26.41
	RotMoLE	**32.92**	**24.35**	**28.02**	**25.37**	**23.17**	**30.21**	**12.50**	**26.76**
2, 4, 2	MoE-LoRA	30.14	26.94	28.04	**27.00**	23.11	30.26	11.67	27.04
	RotMoLE	**32.79**	**26.96**	**28.72**	26.85	**23.39**	**30.68**	**12.48**	**27.74**
4, 5, 3	MoE-LoRA	23.40	24.27	24.76	22.73	18.07	26.14	10.54	22.64
	RotMoLE	**25.28**	**25.48**	**25.75**	**24.90**	**18.89**	**27.82**	**11.67**	**24.06**

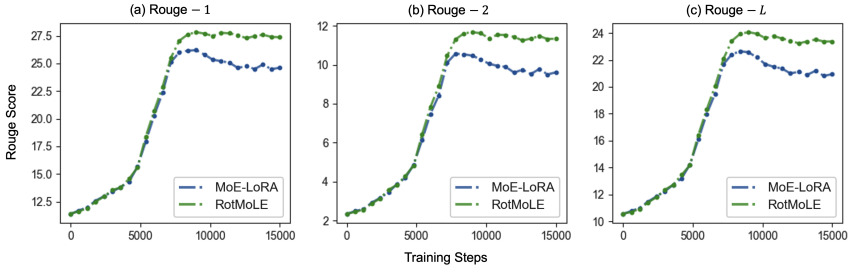


Figure 3: Convergence of Jointly Fine-Tuning on Mixed Multilingual Title Generation(Llama-3.2-3B)

3.3.2 Multilingual NLI

Here we adopt XNLI [Conneau et al., 2018] for our multilingual natural language inference experiments. Appendix C describes the details of XNLI corpus. Notably, the original purpose of XNLI is to evaluate the cross-lingual transfer ability of multilingual models (i.e., whether models purely trained

on English NLI data can handle NLI in other languages). However, in this paper we treat XNLI as a multilingual joint learning dataset to evaluate the performances of models jointly trained on the mixture of samples in five languages. Here we directly set up a sparse MoE and a LoRA expert rank larger than 2, as they are more common scenarios, that is $\{r = 4, n = 5, k = 3\}$. We jointly train the mixture on Llama-3.2-3B for around 4,500 steps until they are fully updated, using a batch size of 50 with 10 per language. The overall and per-language accuracies are illustrated in Table 4. Besides the overall performance, our proposed RotMoLE also outperforms the MoE-LoRA baseline in four out of the five languages. This also indicates that RotMoLE learns more equivalent capacities across different languages during the joint training process.

Table 4: Performance of Jointly Fine-Tuning on Mixed Multilingual NLI (Llama-3.2-3B).

r, n, k	**Method**	**EN**	**ZH**	**ES**	**FR**	**DE**	**Avg.**
4, 5, 3	MoE-LoRA	**52.25**	46.30	48.85	48.90	47.45	48.75
	RotMoLE	51.96	**47.94**	**50.38**	**49.14**	**49.12**	**49.71**

3.4 Comparison with Other Baselines

Although there are many variants proposed for enhancing MoE-LoRA, most of them focus on the loss design during back-propagation, such as the load-balancing loss or the contrastive loss. Considering that our proposed RotMoLE is a gating-based forward method, here we compare our RotMoLE with three variants of MoE-LoRA that specifically focus on improving forward routing effectiveness, including MoCLE [Gou et al., 2023], MoLA [Gao et al., 2024], and AdaMoLE [Liu and Luo, 2024]. MoCLE first clusters the input embeddings into several clusters, and then conducts a top-1 cluster-level expert routing along with a permanently-activated universal expert; MoLA assigns more experts to higher layers given a fixed total number of experts; AdaMoLE replaces the top-k activation by a top-p activation which selects experts whose gate values are larger than a threshold p, and dynamically adjusts p by a network.

We conduct our comparisons on the mixture of QA tasks in Section 3.2.1, and apply Qwen-2.5-3B as our foundation model. Since MoCLE implements a top-1 activation and an extra universal expert, it is distinctive only under the condition of sparse MoE with 2 experts activated. As a result, we set the number of experts n to 4, and the number of activated experts k to 2. Additionally, we naturally set the number of clusters in MoCLE to the number of distinct tasks jointly trained during our experiments, which is 3; For MoLA, we set the expert allocation strategy among the first nine layers of Qwen-2.5-3B to $[2, 2, 2, 4, 4, 4, 6, 6, 6]$, which is equivalent to the configuration of 4 experts per layer in terms of total expert count. Finally, we set the per-expert rank r to 6 to directly test our method under the circumstance of large ranks. Table 5 shows the results. An outperformance of our RotMoLE over the other three MoE-LoRA variants is observed, in terms of both overall and most of per-task accuracies.

Table 5: Performance of Jointly Fine-Tuning on Mixed QA Tasks by Various MoE-LoRA Methods (Qwen-2.5-3B, $r = 6, n = 4, k = 2$).

Method	**CSQA**	**OBQA**	**SIQA**	**Avg.**
MoCLE	36.28	29.60	37.82	34.57
MoLA	39.31	30.00	37.67	35.66
AdaMoLE	**49.39**	37.00	37.87	41.42
RotMoLE (Ours)	43.00	**46.60**	**47.85**	**45.82**

3.5 Rotation Distribution Analysis

We also conduct internal probing experiments for our proposed RotMoLE, to analyze the distribution of rotational biases (angles) allocated to different sub-tasks. Specifically, during our validating experiments on the mixture of QA tasks under Qwen-2.5-3B, we store the per-task rotation bias value θs for the first expert within the v_proj module in the 8-th layer (the last layer with trainable MoE-LoRA adapters). Then we compute the sequence-level rotation bias by averaging the rotation bias values for each token within every validating sequence. Finally, we plot the distribution of

sequence-level rotation biases for each sub-task. Figure 4 exhibits this distribution after 300, 600, 900, and 1,200 training steps, respectively. Here the configuration is $\{r = 2, n = 4, k = 2\}$. It is observed that: (1) During training process, rotation biases become larger and non-zero, and their distributions become more flat and dispersed, indicating our rotations are playing roles in modeling sample diversity, aligning with our illustrations in Figure 2; and (2) The deviation between three sub-task distributions becomes more obvious, indicating our rotations are also playing roles in sub-task specialization and modeling the three QA sub-tasks in different ways.

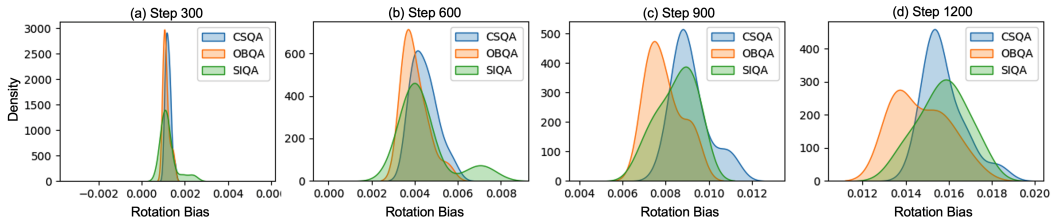


Figure 4: Rotation bias distribution of the first expert within the v_proj module in the 8-th layer.

4 Conclusion

We propose the Rotatable Mixture of Low-rank Experts (i.e., RotMoLE), which is a novel MoE-LoRA framework that enhances the representational capacity of conventional MoE-LoRA in diverse and complicated scenarios through an additional rotational gating mechanism. Unlike conventional MoE approaches that rely solely on scalar scaling operations for expert aggregation, RotMoLE introduces a rotation gate that performs 2D rotations on r-dimensional low-rank space of experts, thereby significantly expanding the solution space of weighted expert combinations. Our extensive experiments across multi-task and multilingual joint learning demonstrate the effectiveness and generalization ability of RotMoLE. Appendix H discusses the limitations.

Acknowledgments and Disclosure of Funding

This research is sponsored by National Key R&D Program of China (2025YFC3409902), NSFC for Distinguished Young Scholar (62425601), New Cornerstone Science Foundation through the XPLORER PRIZE, Beijing Municipal Natural Science Foundation (L251042), National Natural Science Foundation of China (62406036), and CIPS-SMP-Zhipu Large Model Fund (No. CIPS-SMP20250319).

References

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A Survey on Mixture of Experts in Large Language Models. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Jie Cao, Zhenxuan Fan, Zhuonan Wang, Tianwei Lin, Ziyuan Zhao, Rolan Yan, Wenqiao Zhang, Feifei Shao, Hongwei Wang, Jun Xiao, et al. CoMoL: Efficient Mixture of LoRA Experts via Dynamic Core Space Merging. *arXiv preprint arXiv:2603.00573*, 2026.

Yiran Chen, Zhenqiao Song, Xianze Wu, Danqing Wang, Jingjing Xu, Jiaze Chen, Hao Zhou, and Lei Li. MTG: A Benchmark Suite for Multilingual Text Generation. *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2508–2527, 2022.

Ronan Collobert, Samy Bengio, and Yoshua Bengio. A Parallel Mixture of SVMs for Very Large Scale Problems. *Advances in Neural Information Processing Systems*, 14, 2001.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: Evaluating Cross-Lingual Sentence Representations. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, 2018.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, et al. LoRAMoE: Alleviating World Knowledge Forgetting in Large Language Models via MoE-Style Plugin. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945, 2024.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning Factored Representations in a Deep Mixture of Experts. *arXiv e-prints*, pages arXiv–1312, 2013.

Sugyeong Eo, Jung Jun Lee, Chanjun Park, and Heui-Seok Lim. Mixture-of-Clustered-Experts: Advancing Expert Specialization and Generalization in Instruction Tuning. *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14212–14223, 2025.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. Higher Layers Need More LoRA Experts. *arXiv e-prints*, pages arXiv–2402, 2024.

Chendi Ge, Xin Wang, Zeyang Zhang, Hong Chen, Jiapei Fan, Longtao Huang, Hui Xue, and Wenwu Zhu. Dynamic Mixture of Curriculum LoRA Experts for Continual Multimodal Instruction Tuning. *Forty-second International Conference on Machine Learning*, 2025.

Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T Kwok, and Yu Zhang. Mixture of Cluster-Conditional LoRA Experts for Vision-Language Instruction Tuning. *arXiv e-prints*, pages arXiv–2312, 2023.

Hongcan Guo, Guoshun Nan, Yuan Yang, Diyang Zhang, Haotian Li, Zhican Chen, Qinchuan Zhou, Yuhan Ran, Xinye Cao, Sicong Leng, et al. Two is better than one: Rotations scale loras. *arXiv preprint arXiv:2505.23184*, 2025a.

Yongxin Guo, Zhenglin Cheng, Xiaoying Tang, Zhaopeng Tu, and Tao Lin. Dynamic Mixture of Experts: An Auto-Tuning Approach for Efficient Transformer Models. *The Thirteenth International Conference on Learning Representations*, 2025b.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Daniel Fidel Harvey, George Weale, and Berk Yilmaz. Optimizing MoE Routers: Design, Implementation, and Evaluation in Transformer Models. *arXiv preprint arXiv:2506.16419*, 2025.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient Low Rank Adaptation of Large Models. *Proceedings of the 41st International Conference on Machine Learning*, pages 17783–17806, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, et al. LoRA: Low-Rank Adaptation of Large Language Models. *International Conference on Learning Representations*, 1(2):3, 2022.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, 1991.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, et al. Mixtral of Experts. *arXiv e-prints*, pages arXiv–2401, 2024.

Damjan Kalajdzievski. A Rank Stabilization Scaling Factor for Fine-Tuning with LoRA. *arXiv e-prints*, pages arXiv–2312, 2023.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. *International Conference on Learning Representations*, 2021.

Brian Lester, Rami Al-Rfou, and Noah Constant. The Power of Scale For Parameter-Efficient Prompt Tuning. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, 2021.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, et al. MixLoRA: Enhancing Large Language Models Fine-Tuning with LoRA-based Mixture of Experts. *arXiv e-prints*, pages arXiv–2404, 2024.

Xiang Lisa Li and Percy Liang. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *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, 2021.

Chin-Yew Lin. Rouge: A Package for Automatic Evaluation of Summaries. *Text Summarization Branches Out*, pages 74–81, 2004.

Yang Lin, Xinyu Ma, Xu Chu, Yujie Jin, Zhibang Yang, Yasha Wang, and Hong Mei. LoRA Dropout as a Sparsity Regularizer for Overfitting Control. *arXiv e-prints*, pages arXiv–2404, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, et al. DeepSeek-V3 Technical Report. *arXiv e-prints*, pages arXiv–2412, 2024a.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper Than In-Context Learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022a.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. When MoE Meets LLMs: Parameter Efficient Fine-Tuning for Multi-Task Medical Applications. *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1104–1114, 2024b.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-Decomposed Low-Rank Adaptation. *Forty-first International Conference on Machine Learning*, 2024c.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-Tuning: Prompt Tuning Can Be Comparable to Fine-Tuning Across Scales and Tasks. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, 2022b.

Zefang Liu and Jiahua Luo. AdaMoLE: Fine-Tuning Large Language Models with Adaptive Mixture of Low-Rank Adaptation Experts. *First Conference on Language Modeling*, 2024.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. MoELoRA: Contrastive Learning Guided Mixture of Experts on Parameter-Efficient Fine-Tuning for Large Language Models. *arXiv e-prints*, pages arXiv–2402, 2024.

Yingjie Ma, Zitong Yu, Xun Lin, Weicheng Xie, and Linlin Shen. BIG-MoE: Bypass Isolated Gating MoE for Generalized Multimodal Face Anti-Spoofing. *arXiv e-prints*, pages arXiv–2412, 2024.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models. *Advances in Neural Information Processing Systems*, 37:121038–121072, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, 2018.

Lin Mu, Haiyang Wang, Li Ni, Lei Sang, Zhize Wu, Peiquan Jin, and Yiwen Zhang. TalkLoRA: Communication-Aware Mixture of Low-Rank Adaptation for Large Language Models. *arXiv preprint arXiv:2604.06291*, 2026.

Xiaonan Nie, Xupeng Miao, Shijie Cao, Lingxiao Ma, Qibin Liu, Jilong Xue, Youshan Miao, Yi Liu, Zhi Yang, and Bin Cui. EvoMoE: An Evolutional Mixture-of-Experts Training Framework via Dense-To-Sparse Gate. *arXiv e-prints*, pages arXiv–2112, 2021.

Ruizhong Qiu, Hanqing Zeng, Yinglong Xia, Yiwen Meng, Ren Chen, Jiarui Feng, Dongqi Fu, Qifan Wang, Jiayi Liu, Jun Xiao, et al. ReMix: Reinforcement Routing for Mixtures of LoRAs in LLM Finetuning. *arXiv preprint arXiv:2603.10160*, 2026.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

Xiaozhe Ren, Pingyi Zhou, Xinfan Meng, Xinjing Huang, Yadao Wang, Weichao Wang, et al. PanGu-{\Sigma}: Towards Trillion Parameter Language Model with Sparse Heterogeneous Computing. *arXiv e-prints*, pages arXiv–2303, 2023.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQA: Commonsense Reasoning About Social Interactions. *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, 2019.

Shrey Shah and Justin Wagle. MoLoRA: Composable Specialization via Per-Token Adapter Routing. *arXiv preprint arXiv:2603.15965*, 2026.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. *International Conference on Learning Representations*, 2017.

Boyan Shi, Wei Chen, Shuyuan Zhao, Junfeng Shen, Shengnan Guo, Shaojiang Wang, and Huaiyu Wan. SAMoRA: Semantic-Aware Mixture of LoRA Experts for Task-Adaptive Learning. *arXiv preprint arXiv:2604.19048*, 2026.

Mengyang Sun, Yihao Wang, Tao Feng, Dan Zhang, Yifan Zhu, and Jie Tang. A Stronger Mixture of Low-Rank Experts for Fine-Tuning Foundation Models. *International Conference on Machine Learning*, pages 57712–57727, 2025.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. *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 4149–4158, 2019.

GLM Team, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, et al. ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools. *arXiv e-prints*, pages arXiv–2406, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, et al. LLaMA: Open and Efficient Foundation Language Models. *arXiv e-prints*, pages arXiv–2302, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv e-prints*, pages arXiv–2307, 2023b.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. MiLoRA: Harnessing Minor Singular Components for Parameter-Efficient LLM Finetuning. *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4823–4836, 2025a.

Shaowen Wang, Linxi Yu, and Jian Li. LoRA-GA: Low-Rank Adaptation with Gradient Approximation. *Advances in Neural Information Processing Systems*, 37:54905–54931, 2024a.

Sheng Wang, Liheng Chen, Jiyue Jiang, Boyang Xue, Lingpeng Kong, and Chuan Wu. LoRA Meets Dropout Under A Unified Framework. *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1995–2008, 2024b.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan, and Jianfeng Gao. AdaMix: Mixture-of-Adaptations for Parameter-Efficient Model Tuning. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5744–5760, 2022.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. LoRA-Pro: Are Low-Rank Adapters Properly Optimized? *The Thirteenth International Conference on Learning Representations*, 2025b.

Shaohua Wu, Jiangang Luo, Xi Chen, Lingjun Li, Xudong Zhao, Tong Yu, Chao Wang, et al. Yuan 2.0-M32: Mixture of Experts with Attention Router. *arXiv e-prints*, pages arXiv–2405, 2024a.

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

Lei Xu, Michael Jordan, and Geoffrey E Hinton. An Alternative Model for Mixtures of Experts. *Advances in Neural Information Processing Systems*, 7, 1994.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, et al. Qwen2.5 Technical Report. *arXiv e-prints*, pages arXiv–2412, 2024a.

Shu Yang, Muhammad Asif Ali, Cheng-Long Wang, Lijie Hu, and Di Wang. MoRAL: MoE Augmented LoRA for LLMs' Lifelong Learning. *arXiv e-prints*, pages arXiv–2402, 2024b.

Seniha Esen Yuksel, Joseph N Wilson, and Paul D Gader. Twenty Years of Mixture of Experts. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1177–1193, 2012.

Dan Zhang, Tao Feng, Lilong Xue, Yuandong Wang, Yuxiao Dong, and Jie Tang. Parameter-Efficient Fine-Tuning for Foundation Models. *arXiv e-prints*, pages arXiv–2501, 2025a.

Danyang Zhang, Junhao Song, Ziqian Bi, Yingfang Yuan, Tianyang Wang, Joe Yeong, and Junfeng Hao. Mixture of Experts in Large Language Models. *arXiv e-prints*, pages arXiv–2507, 2025b.

Fangzhao Zhang and Mert Pilanci. Riemannian Preconditioned LoRA for Fine-Tuning Foundation Models. *Proceedings of the 41st International Conference on Machine Learning*, pages 59641–59669, 2024.

Xu Zhang, Kaidi Xu, Ziqing Hu, and Ren Wang. Optimizing Robustness and Accuracy in Mixture of Experts: A Dual-Model Approach. *International Conference on Machine Learning*, pages 76436–76450, 2025c.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, et al. A Survey of Large Language Models. *arXiv e-prints*, pages arXiv–2303, 2023.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, et al. Mixture-of-Experts with Expert Choice Routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

Yun Zhu, Nevan Wichers, Chu-Cheng Lin, Xinyi Wang, Tianlong Chen, Lei Shu, Han Lu, et al. SiRA: Sparse Mixture of Low Rank Adaptation. *arXiv e-prints*, pages arXiv–2311, 2023.

A Related Work

A.1 Mixture of Experts in LLMs

The MoE architecture, known as a framework for efficiently increasing model capacity without significant computational overhead, consists of a series of specialized modules called *experts*, alongside a routing module known as *gate*. In the concept of MoE, each expert is dedicated to a specific domain or task, or granular features of the data [Cai et al., 2025]. This fundamental idea was initially introduced by Jacobs et al. [1991] to alleviate the issue of cross-interference between tasks and enhance training efficiency. Subsequent studies, such as Xu et al. [1994], Collobert et al. [2001], Yuksel et al. [2012], and Eigen et al. [2013], have all made impressive contributions to the field of MoE. Depending on how the gating module is implemented, MoE architectures can be basically separated into two types: dense MoE and sparse MoE.

Dense MoE implements a soft gating mechanism that activates all existing experts during both training and inference. The output can be described as $\sum_{i=1}^{N_{expert}} g_i E_i(x)$, where $E_i(x)$ denotes the output of expert i, and g_i is the corresponding gate value. To align with the statistical mixture theory and preserve signal magnitude, MoE usually constrains the sum of gate values to 1.0 by applying a *Softmax* function for normalization, such that $g = Softmax(x \cdot W_g)$; In contrast, sparse MoE was subsequently proposed by Shazeer et al. [2017] to sparsely activate a small group of experts during each forward propagation, given by $y = \sum_{i \in TopK(x)} g_i E_i(x)$ where $TopK(x)$ denotes the set of K selected experts with the highest gate values. This strategy achieves better computational efficiency through its sparse activation while still benefiting from a large pool of candidate experts with extensive knowledge. Gshard [Lepikhin et al., 2021] was the first to introduce the sparse MoE architecture to Transformer models, after which this powerful and flexible design has begun to be increasingly adopted for training large language models. Llama [Touvron et al., 2023a], DeepSeek [Liu et al., 2024a], Mixtral [Jiang et al., 2024], and many other mainstream LLMs are trained based on sparse MoE architectures.

However, despite the success of MoE, some researchers have proposed that the routing mechanisms inside MoE may act as a bottleneck of the whole structure. For example, Zhang et al. [2025c] demonstrates that conventional MoE architectures exhibit sudden performance drops during training epochs due to the router's sensitivity to input perturbations; Besides, Ma et al. [2024] emphasizes that conventional gating networks suffer from noise-sensitive and susceptible decision-making that cannot capture nuanced input characteristics. Therefore, some existing studies also focus on boosting the routing ability of MoE. For example, Eo et al. [2025] applies a dual-stage routing mechanism consisting of a sentence-level group routing and a token-level expert routing within the selected group; Guo et al. [2025b] implements a dynamic mechanism to determine the number of selected experts k instead of treating it as a predefined hyper-parameter; To alleviate the routing imbalance, Zhou et al. [2022] enables experts to select top-k tokens instead of enabling tokens to select experts, allowing each expert to maintain a fixed bucket size; Wu et al. [2024a] designs an attention mechanism for routing, which treats token embeddings as queries and expert embeddings as keys; Harvey et al. [2025] conducts a comparison among six MoE routing mechanisms and concludes that simple routers like a linear module may suffer from overfitting; while complex routers like MLP may suffer from low certainty and thus constrain expert specialization. However, most studies still solely rely on the context of enhancing the scaling routers and their gate value distributions, lacking attention to more complex expert transformations.

A.2 PEFT and LoRA

PEFT [Han et al., 2024] is a family of techniques designed for adapting large pre-trained language models to downstream tasks while avoiding the prohibitive cost of fully fine-tuning. Instead of updating all billions of parameters, PEFT methods freeze most of the original model weights and only update a small number of trainable parameters or modules, dramatically reducing computational and memory requirements. Popular PEFT methods include Prompt Tuning [Lester et al., 2021], Prefix Tuning [Li and Liang, 2021], P-Tuning V2 [Liu et al., 2022b], $(IA)^3$ [Liu et al., 2022a], and LoRA [Hu et al., 2022], etc. Among these approaches, LoRA has emerged as one of the most influential and widely adopted methods due to its simplicity, effectiveness, and theoretical elegance. Specifically, LoRA approximates the full-rank weight updates $\Delta W_{d\times d}$ through decomposing it into a product of two low-rank trainable matrices $B_{d\times r}$ and $A_{r\times d}$, formulated as $W = W_0 + \Delta W = W_0 + BA$.

Here, the rank r is much smaller than the weight matrix dimension d. As a result, LoRA significantly reduces the number of trainable parameters.

To enhance LoRA, researchers have explored several technical directions. These include: (1) Improving the initialization of LoRA modules, such as PISSA [Meng et al., 2024], MiLoRA [Wang et al., 2025a], and DoRA [Liu et al., 2024c]; (2) Optimizing the gradient updates of LoRA, such as LoRA-Pro [Wang et al., 2025b], LoRA-GA [Wang et al., 2024a], Riemannian Preconditioned LoRA [Zhang and Pilanci, 2024], and LoRA+ [Hayou et al., 2024]; (3) Incorporating regularization and dropout techniques within LoRA, such as RsLoRA [Kalajdzievski, 2023], HiddenKey [Wang et al., 2024b], and LoRA Dropout [Lin et al., 2024]; and (4) Integrating multiple LoRA modules into a mixture framework, known as the Mixture of LoRAs or MoE-LoRA, as proposed by Dou et al. [2024], Gou et al. [2023], Gao et al. [2024], Luo et al. [2024], Liu and Luo [2024], and Sun et al. [2025]. In general, these diverse approaches all aim to address the performance limitation of conventional LoRA through bridging the gap with fully fine-tuning while still maintaining parameter efficiency.

A.3 MoE-LoRA

Inspired by Wang et al. [2022], many researchers have contributed to the field of MoE-LoRA. For example, LoRAMoE [Dou et al., 2024] designs a MoE-LoRA framework for alleviating world knowledge forgetting during fine-tuning LLMs by allocating some experts to specialize only in exploiting pretrained knowledge for task solving; To overcome task conflicts, MoCLE [Gou et al., 2023] first groups all instructions and then conducts cluster-level expert routing along with a universal expert; MoLA [Gao et al., 2024] assigns a varying number of experts to different layers and emphasizes that higher layers need more LoRA experts; Luo et al. [2024] introduces a contrastive loss into MoE-LoRA to encourage LoRA experts to learn distinct features and improve the efficiency of expert specialization; AdaMoLE [Liu and Luo, 2024] applies a dynamic activation threshold p for each expert and implements a learnable network to adjust it; Instead of training from scratch, MoLE [Wu et al., 2024b] discusses the scenario of integrating multiple existing LoRA modules that are already trained on different data sources, and proposes an MoE framework to merge them; Sun et al. [2025] proposes an algorithm to integrate Riemannian preconditioners into MoE-LoRA for an enhanced training efficiency and effectiveness; MoRAL [Yang et al., 2024b] focuses on implementing MoE-LoRA for continual lifelong learning; Liu et al. [2024b] focuses on applying MoE-LoRA in multi-task medical applications; MoLoRA [Shah and Wagle, 2026] focuses on per-token adapter routing; TalkLoRA [Mu et al., 2026] introduces a talking module to low-rank experts for information exchange; CoMoL [Cao et al., 2026] proposes a core matrix for LoRA, and only implements a routing structure on multiple core matrices instead of multiple low-rank pairs; ReMix [Qiu et al., 2026] learns routing strategies through reinforcement learning to alleviate the routing weights collapse; SAMoRA [Shi et al., 2026] uses a semantic-aware routing method in MoE-LoRA for task-adaptive learning; Works such as Dou et al. [2024], Li et al. [2024], and Zhu et al. [2023] focus on load balancing techniques for low-rank experts; Among these works, RadarGate Guo et al. [2025a] is the first to introduce rotation mechanism into MoE-LoRA routing. Specifically, it assigns each LoRA expert a d-dimensional rotation matrix constructed by a RoPE-style combination of multiple 2-dimensional rotation matrices. Their rotation angles are generated from the input and the relative relationships among LoRA experts via a learnable projection module.

All the existing MoE-LoRA works can be roughly separated into two categories: (1) Improving MoE-LoRA by innovative architectural or algorithmic designs; and (2) Applying the MoE-LoRA framework to some specific domains to solve complex multi-task problems. However, none of the works have noticed that the low-rank feature in each LoRA expert can be fully utilized for enhancing MoE-LoRA capabilities through implementing an efficient low-rank transformation for each selected expert beyond the basic gate-value-based scaling in conventional MoE.

B Details of Overall Experimental Setup

In most of our experiments, we mainly adopt Llama-3.2-3B [Touvron et al., 2023b] as our foundation model, while for some experiments we also employ Qwen-2.5-3B [Yang et al., 2024a] and GLM-4-9B [Team et al., 2024] to demonstrate the generality of our method. We fine-tune all the models on the NVIDIA A100 80GB GPU. To accelerate experiments and reduce resource consumption, we

integrate MoE-LoRA trainable modules only into the q_proj, k_proj, v_proj, and o_proj modules **only from the first quarter of the model layers**. Specifically, for the 28-layer Llama-3.2-3B, we train the first seven layers (i.e., layer 0-6); For the 36-layer Qwen-2.5-3B, we train the first nine layers (i.e., layer 0-8); While for the 40-layer GLM-4-9B, we train the first ten layers (i.e., layer 0-9). During training, we use a stochastic gradient descent optimizer and set the initial learning rate to 3e-4 with a linear decay scheduler. We fix the system random seed for each run to eliminate the impact of randomization and ensure reproducibility. For initialization, we follow the convention of LoRA to initialize module B to 0. We also initialize our rotation gate W_θ to 0, and implement a Kaiming uniform initialization for the trainable center vectors $\boldsymbol{q}$.

To facilitate joint learning, we equally distribute the number of samples across all tasks within every batch. Since the size of the training set varies across different tasks, we choose to **train each task for the same number of steps rather than epochs**. Consequently, all tasks are treated equally regardless of their difficulty. This setting increases the complexity to conduct joint learning on LLMs, as simple and complicated tasks may exhibit different converging speeds — when simple tasks show signs of overfitting, complicated tasks may remain unconverged.

C Dataset Overview

Mixture of QA Tasks: Our mixture of QA tasks includes CommonsenseQA [Talmor et al., 2019], OpenBookQA [Mihaylov et al., 2018], and SocialIQA [Sap et al., 2019]. Specifically, CommonsenseQA consists of more than 12,000 questions requiring multiple types of common sense knowledge; OpenBookQA consists of roughly 6,000 questions from real open book exams on elementary level science facts; while SocialIQA consists of around 38,000 questions about emotional and social reactions during people's everyday life.

Mixture of GLUE Tasks: The GLUE benchmark [Wang et al., 2018] targets for evaluating model performances on natural language understanding. It includes nine diverse NLU tasks varying from sentiment classification to grammar checking. To simplify our experiments, here we only select three of the tasks as the mixture: (1) SST-2 which is a sentiment classification task for movie reviews; (2) MRPC which is a paraphrase detection task that determines whether two sentences are semantically equivalent; and (3) QNLI which is a verifying task for candidate QA pairs derived from SQuAD [Rajpurkar et al., 2016].

Mixture of MTG Title Generation Tasks: The MTG (Multilingual Text Generation) benchmark [Chen et al., 2022] is a human-annotated multilingual text generation benchmark designed for fine-tuning and evaluating language models across diverse languages and tasks. It consists of 400,000 human-annotated samples covering four distinct generation tasks, including the title generation task across five languages: English, Chinese, Spanish, French, and German. Its title generation task, sourced from the ByteCup news corpus, requires models to convert a given article into a condensed, faithful title that preserves the main idea while encouraging readability.

Mixture of XNLI Tasks: XNLI [Conneau et al., 2018], known as a cross-lingual version of the Natural Language Inference (NLI) corpus, involves 7,500 human-annotated examples in 15 languages, such as English, Chinese, Spanish, etc. Specifically, every sample in XNLI contains a pair of premise and hypothesis sentences, along with a label indicating if the hypothesis logically follows, contradicts, or is irrelevant to the premise. Notably, to facilitate the experiments and align with MTG's experimental settings, we select the five MTG languages — English, Chinese, Spanish, French, and German — from all the 15 languages in XNLI as the language mixture.

D Convergence Performance on the Mixture of QA Tasks

During our fine-tuning experiments on the mixture of QA tasks in Section 3.2.1, we also analyze the converging performances of both the MoE-LoRA baseline and our proposed RotMoLE. Theoretically speaking, RotMoLE implements more trainable parameters and should therefore be more difficult to converge compared to the MoE-LoRA baseline with only a scaling gate. However, experimental results indicate that our RotMoLE does not converge more slowly; while in some cases, it is even faster. Figure 5 exhibits this phenomenon. We consider it as an advantage of the rotation gate: using both scaling and rotation to model experts' exploitation is much easier than using scaling

alone. Furthermore, RotMoLE also exhibits better converging stability and anti-overfitting ability in experiments under configuration $\{r = 2, n = 4, k = 2\}$ and $\{r = 3, n = 2, k = 2\}$.

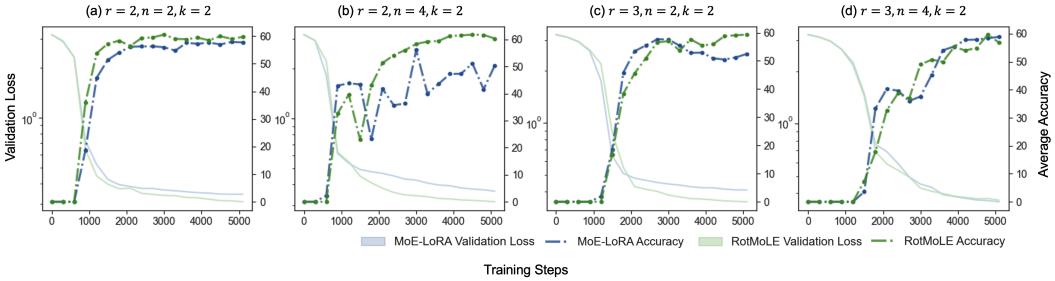


Figure 5: Convergence of Jointly Fine-Tuning on Mixed QA Tasks (Qwen-2.5-3B).

E Ablation Study

E.1 Impact of Trainable Size

Although our method demonstrates a comprehensive improvement over the MoE-LoRA baselines across various scenarios, these gains are built upon the introduction of additional rotation gates $\theta = \Theta(x)$ and center vectors $\boldsymbol{q}$. This raises the question of whether our achieved improvements are due to the increase of trainable parameter counts, rather than our novel rotary architecture design. To clarify this issue, we conduct an experiment on a size-equivalent scaling baseline compared to our proposed RotMoLE. Specifically, we introduce more parameters to the conventional scaling gate, and replace its linear module by a size-equivalent two-layer MLP whose hidden dimension is denoted as h. Then we have:

$$2dn + rn = dH + Hn \Rightarrow H = \frac{2dn + rn}{d + n} \approx 2n,$$

considering that $r \ll d$ and $n \ll d$. Consequently, an MLP with a hidden dimension $2n$ serves as the scaling gate of the size-equivalent baseline. We test this size-equivalent baseline under the mixture of three GLUE tasks, which is already elaborated in Section 3.2.2. We use the configuration $\{r = 4, n = 5, k = 3\}$ and present the performance comparisons among the conventional MoE-LoRA baseline, the size-equivalent baseline, and our proposed RotMoLE in Table 6. All other settings are the same as in Section 3.2.2. It is observed that the MLP gate performs even worse than the MoE-LoRA baseline with a linear gate, only achieving the best for the sub-task SST-2. We consider it as a phenomenon of overfitting which leads to an unbalanced performance. To illustrate it, we plot the converging traces of MoE-LoRA$_{MLP}$ and RotMoLE in Figure 6 from step 1,500. Figure 6 indicates that the sub-task SST-2 has already converged, while the other two are still improving. This usually means that SST-2 is much easier than the other two sub-tasks, especially the sub-task QNLI which still has much potential for improvement. MoE-LoRA$_{MLP}$ is overfitted to the easier sub-task SST-2 while lacking sufficient learning of the opportunistic QNLI. This confirms our statement that the MLP gate module is more likely to cause an unbalanced performance and fail to fully utilize the potential of its additional parameters.

Table 6: Performance of Jointly Fine-Tuning on Mixed GLUE Tasks by the MoE-LoRA Baseline, the Size-Equivalent MoE-LoRA with MLP Gates, and RotMoLE (Llama-3.2-3B, $r = 4, n = 5, k = 3$).

Method	**SST-2**	**MRPC**	**QNLI**	**Avg.**
MoE-LoRA	88.76	69.19	61.63	73.19
MoE-LoRA$_{MLP}$	**89.56**	69.16	58.19	72.30
RotMoLE	89.22	**69.21**	**61.83**	**73.42**

E.2 Adapted Layers

In Section 3, we mainly deploy LoRA adapters on the first quarter of layers just for simplicity, and there is no special reason for that. Here we conduct ablation study on a different part of layers using

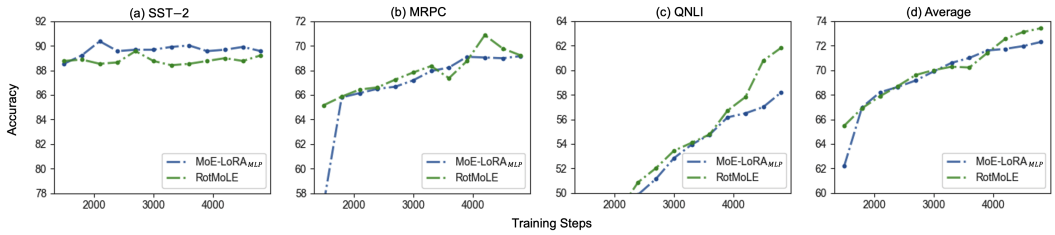


Figure 6: Convergence of Jointly Fine-Tuning on Mixed GLUE Tasks by both MoE-LoRA$_{MLP}$ and RotMoLE (Llama-3.2-3B, $r = 4, n = 5, k = 3$).

a different model to verify the universality of our method. Specifically, here we use Qwen-2.5-14B as our foundation model, which has 48 layers. The middle 12 layers (Layer 18-29) of Qwen-2.5-14B are adapted by MoE-LoRA adapters. We set the MoE configuration to $\{r = 4, n = 5, k = 3\}$, and all the other settings are the same as Section 3.2.1. The experimental results of fine-tuning model on mixed QA tasks are listed in Table 7. These results can also be used to verify the effectiveness of RotMoLE on larger models, which Appendix E.4 discusses about.

Table 7: Performance of Jointly Fine-Tuning on Mixed QA Tasks under Qwen-2.5-14B. Layer 18-29 are adapted by MoE-LoRA and RotMoLE adapters ($r = 4, n = 5, k = 3$).

Method	**CSQA**	**OBQA**	**SIQA**	**Avg.**
MoE-LoRA	76.25	79.20	73.90	76.45
RotMoLE	**77.07**	**81.60**	**74.77**	**77.81**

E.3 Effectiveness of Components

Here we conduct experiments for ablation study of different components in our RotMoLE, which are the rotation gate Θ and the center vectors $\boldsymbol{q}$. Specifically, since $\boldsymbol{q}$ and θ logically require each other during forwarding in our proposed structure, we can not singly remove one of them. Instead, we respectively set the learning rate of each to 0, to eliminate the influence of one for ablation study on the other one. We test on the mixed QA tasks, and results are listed in Table 8. Here Qwen-2.5-3B serves as the foundation model. The MoE configuration is $\{r = 4, n = 5, k = 3\}$, and all the other settings are the same as Section 3.2.1.

Table 8: Performance of Jointly Fine-Tuning on Mixed QA Tasks by MoE-LoRA Baseline and RotMoLE Variants for Ablation Study on Θ and $\boldsymbol{q}$ (Qwen-2.5-3B, $r = 4, n = 5, k = 3$).

Method	**Average Accuracy**
MoE-LoRA	46.91
RotMoLE without training rotation gate Θ	44.67
RotMoLE without training center vectors $\boldsymbol{q}$	52.53
RotMoLE	**54.03**

E.4 Base Model Size

We have already examined the effectiveness of our proposal in previous sections by conducting multi-task and multilingual joint learning experiments under Llama-3.2-3B and Qwen-2.5-3B. Now we turn to another foundation model of a different model size, GLM-4-9B. We follow all the experimental settings of Section 3.2.1 to conduct comparable experiments of the QA mixture under GLM-4-9B. What is different is that we set the initial learning rate to 9e-4, and train the model for 8,000 steps. The results are shown in Table 9. An outperformance of our RotMoLE is still observed in both overall and per-task accuracy.

Table 9: Performance of Jointly Fine-Tuning on Mixed QA Tasks (GLM-4-9B).

r	n, k	Method	CSQA	OBQA	SIQA	Avg.
2	2, 2	MoE-LoRA	50.37	40.20	45.70	45.42
		RotMoLE	**57.33**	**47.00**	**52.35**	**52.23**
	4, 2	MoE-LoRA	48.32	26.80	42.78	39.30
		RotMoLE	**57.82**	**51.80**	**52.71**	**54.11**
3	2, 2	MoE-LoRA	49.55	44.00	49.74	47.76
		RotMoLE	**68.06**	**66.20**	**68.01**	**67.42**
	4, 2	MoE-LoRA	57.25	50.60	55.07	54.30
		RotMoLE	**77.48**	**65.60**	**72.16**	**71.75**

F Efficiency Overhead

Here we report the inference latency overhead of our proposed RotMoLE. Specifically, experiments are conducted under both Qwen-2.5-3B and Qwen-2.5-14B base models to demonstrate how the rotation overhead scales with model size. For simplicity, we cut the length of all input sequences to $1k$, and set the MoE configuration to $\{r = 4, n = 5, k = 3\}$. For both the 3B and 14B models, Layer 0-8 are adapted by MoE-LoRA adapters. Table 10 shows that, although RotMoLE exhibits more latency overhead than MoE-LoRA baseline, compared to the whole inference time, the overhead rate is well restricted.

Table 10: Inference Latency of MoE-LoRA Baseline and RotMoLE ($r = 4, n = 5, k = 3$).

Batch	Model	Inference Latency (ms)		Overhead (ms)
		MoE-LoRA	RotMoLE	
4	Qwen-2.5-3B	368.73	390.14	+21.41 (+5.81%)
	Qwen-2.5-14B	1375.63	1391.82	+16.19 (+1.18%)
8	Qwen-2.5-3B	684.67	706.10	+21.43 (+3.13%)
	Qwen-2.5-14B	2621.31	2644.33	+23.02 (+0.88%)

G Statistical Significance

Here we report our detailed experimental results for computing statistical significance. Specifically, we run the fine-tuning experiments of mixed QA tasks under Qwen-2.5-3B for three times under differently shuffled initial sample orders (the order before running torch dataset sampler), and calculate the mean and standard error of their average performance over tasks. The MoE configuration is also set to $\{r = 4, n = 5, k = 3\}$ with other configurations remaining the same as Section 3.2.1. Table 11 illustrates the detailed results. It can be witnessed that, although our proposed RotMoLE exhibits larger error bars than MoE-LoRA baseline, it statistically outperforms the baseline even considering those error bars. In addition to mixed QA experiments under Qwen-2.5-3B, we also conduct significance tests for mixed multilingual title generation experiments under Llama-3.2-3B, whose detailed results are illustrated in Table 12.

Table 11: Performance with Error Bars of Jointly Fine-Tuning on Mixed QA Tasks (Qwen-2.5-3B, $r = 4, n = 5, k = 3$).

Method	CSQA	OBQA	SIQA	Avg.
MoE-LoRA	45.81 ± 0.78	39.80 ± 0.59	45.63 ± 0.48	43.75 ± 0.29
RotMoLE	**58.80** ± 1.56	**48.27** ± 3.62	**52.47** ± 4.13	**53.18** ± 2.97

Table 12: Performance with Error Bars of Jointly Fine-Tuning on Mixed Multilingual Title Generation (Llama-3.2-3B, $r = 4, n = 5, k = 3$).

Method	**Avg. Rouge-1**	**Avg. Rouge-2**	**Avg. Rouge-L**
MoE-LoRA	25.57 ± 0.67	10.24 ± 0.34	22.06 ± 0.56
RotMoLE	**27.24** ± 0.70	**11.34** ± 0.38	**23.50** ± 0.52

H Limitations

While our RotMoLE demonstrates significant improvements, we still acknowledge some limitations. For example, our evaluations solely focus on the supervised fine-tuning (SFT) scenarios to validate the efficacy of RotMoLE, not incorporating reinforcement learning methods, such as GRPO, in this paper; Secondly, as we point out the phenomenon that various tasks lead to their specific distributions of rotation biases, a further analysis is preferable on whether/how the distribution distinctions are related to the similarities between tasks.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper's contributions and scope are mainly on integrating a rotation gate into MoE structure to boost the diversity of expert exploitation, which is correctly claimed in the abstract and introduction (Section 1).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of this work are discussed in Appendix H.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: For the theoretical foundation illustrated in Section 2.1, we provide a proof after it in the same section; For our rotation algorithm under larger ranks proposed in Section 2.4, we provide a reasonable assumption at the beginning of the section and then derive the algorithm based on it.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All required experimental details for reproducibility are described in Section 3 and from Appendix B to E.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release our code on Github.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All required training and test details are described in Section 3 and from Appendix B to E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We run experiments many times by varying the hyper-parameters and experimental configurations, which can be considered as appropriately indicating the statistical significance of our out-performance over MoE-LoRA baselines. For experiments targeting exact statistical significances, please refer to Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Information of experimental computing resources are provided in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work is a foundational research and not tied to particular applications.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the original papers of models and datasets are properly cited throughout this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We will release new assets (e.g., the open-sourced code) once the paper is accepted and published. During reviewing, we do not release any new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We design our method for fine-tuning LLMs. Section 3 describes our usage of LLMs as testing objectives and base context for evaluating our proposed algorithm. For specific models mentioned, please refer to Appendix B.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.