GRAPHMOE: Amplifying Cognitive Depth of Mixture-of-Experts Network via Introducing Self-Rethinking Mechanism

Anonymous ACL submission

Abstract

Traditional Mixture-of-Experts (MoE) networks benefit from utilizing multiple smaller expert models as opposed to a single large network. However, these experts typically operate independently, leaving a question open about whether interconnecting these models could enhance the performance of MoE networks. In response, we introduce GRAPHMOE, aiming at augmenting the cognitive depth of language models via a self-rethinking mechanism constructed on Pseudo Graph MoE networks. GRAPHMOE employs a recurrent routing strategy, allowing the model to iteratively revisit and refine intermediate reasoning states. Simultaneously, by treating experts as interconnected nodes in a pseudo-graph, it facilitates iterative information exchange among experts rather than treating them as isolated modules. We implement the GRAPHMOE architecture using Low-Rank Adaptation techniques (LoRA) and conduct extensive experiments on various benchmark datasets. The experimental results reveal that GRAPHMOE outperforms other LoRA based models, achieving state-of-the-art performance.

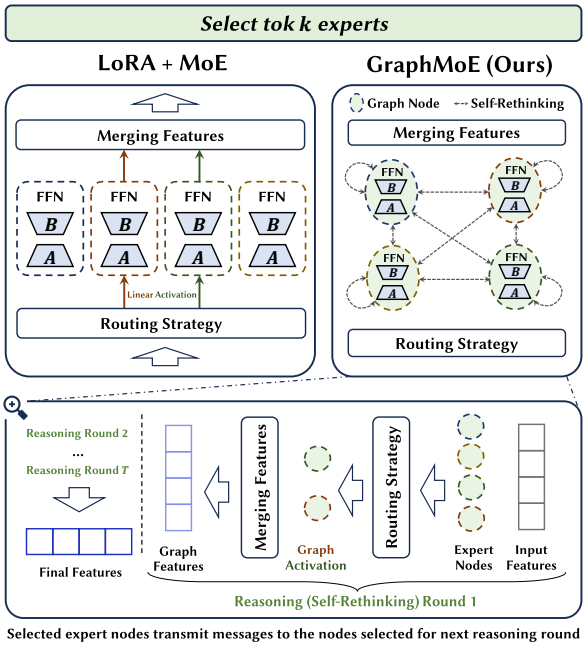


Figure 1: Comparison between LoRA+MoE and GRAPHMOE architectures.

1 Introduction

Large Language Models (LLMs), exemplified by the GPT series, have dramatically advanced the field of Natural Language Processing (NLP), demonstrating exceptional performance across a variety of tasks including commonsense reasoning (Yang et al., 2024), creative generation (Loakman et al., 2023), dialogue generation (Lv et al., 2024; Zhao et al., 2023, 2024b), and summarization (Goldsack et al., 2023; Zhao et al., 2024a). In recent years, there has been a growing interest in refining LLM architectures to deliver superior performance with reduced parameters and lower GPU memory consumption (Tang et al., 2023). A prominent strategy in this endeavor is the Mixture-of-Experts (MoE) architecture (Cai et al., 2024), which pre-trains multiple Feed-Forward Networks (FFNs) as specialized experts, activating a select few during inference. This strategy allows experts to operate optimally within their domains of expertise, often providing enhanced performance compared to LLMs with a similar number of activated parameters.

Despite their effectiveness, existing MoE architectures typically employ experts independently under a linear routing strategy. Since individual experts are optimized for distinct input distributions, such isolated utilization may limit their collective problem-solving potential. We argue that fostering collaboration among experts, interconnected nodes in a graph, could further enhance MoE reasoning capacity. Inspired by knowledge aggregation mechanisms in pseudo-graphs (Tang et al., 2023; Zequn Sun, 2020), we consider a routing paradigm in which expert models are treated as graph nodes that interact cyclically. We hypothesize that, as graphical message passing among expert nodes increases, MoE outputs can progressively narrow the semantic gap with respect to human references.

Based on this motivation, we introduce GRAPHMOE (Graph Mixture of Experts), an MoE framework that enhances the cognitive capabilities of language models by integrating a self-rethinking mechanism into a pseudo-graph-structured expert network. GRAPHMOE emulates human-like iterative interaction by allowing the model to continuously revisit and refine its comprehension of input data through multiple reasoning cycles (Ding et al., 2019; Yan et al., 2023). This is accomplished by implementing a recurrent routing strategy on a pseudo-graph formed by expert models. These models transmit their outputs as signals for selecting subsequent expert batches and as inputs encapsulating aggregated graph features from prior reasoning rounds—a mechanism we term the "self-rethinking mechanism". Given the challenges inherent in pre-training a MoE LLM from scratch, we chose to implement the GRAPHMOE architecture utilizing Low-Rank Adaptation (LoRA) techniques.

In Figure 1, we provide an illustration of GRAPHMOE(LoRA) and compare it with the conventional LoRA+MoE framework. In standard LoRA+MoE approaches, the features of activated expert models are integrated directly. This process renders other inactive expert models non-contributory to problem-solving, particularly when the model tends to consistently select a specific set of expert models, as evidenced by low workload balancing. In contrast, GRAPHMOE(LoRA) conceptualizes expert nodes as graph nodes, enabling message transmission across these interconnected nodes through graph activation over multiple reasoning rounds. This approach maintains the reduced memory usage characteristic of conventional LoRA+MoE models (with same activation number of expert models in each cycle) while increasing computational iterations. Experimental results demonstrate the effectiveness of GRAPHMOE, showing consistent improvements over LoRA-based baselines and achieving state-of-the-art performance across evaluated benchmarks.

The main contributions of this work are as follows:

- We introduce a self-rethinking mechanism that enables MoE models to iteratively revisit and refine intermediate representations, facilitating human-like iterative reasoning.
- We present a pseudo-graph-based MoE architecture that models experts as interconnected nodes, enabling iterative information exchange across experts rather than treating them as independent modules.
- Extensive experiments on a diverse range of benchmarks have been conducted to validate the effectiveness of GRAPHMOE, providing empirical insights into expert collaboration in LLMs.

2 Related Work

The concept of Mixture of Experts (MoE) was first introduced by Jacobs et al. (1991), who proposed training multiple networks (experts) on different subsets of data and aggregating their outputs. Recently, as LLMs have become a focal point of research, MoE layers have been integrated into Transformer-based architectures. Specifically, researchers have replaced standard Feed-Forward Networks (FFNs) with sparse MoE layers, employing novel routing strategies (Zuo et al., 2021; Zhong et al., 2024; Wu et al., 2024a; Muqeeth et al., 2023; Fu et al., 2024) and advanced expert segmentation techniques (Dai et al., 2024; He, 2024; Jiang et al., 2024; Xiao et al., 2024).

Numerous studies have investigated the application of Parameter-Efficient Fine-Tuning (PEFT) methods to introduce additional trainable parameters for implementing pseudo MoE structures within LLMs (Dou et al., 2024; Luo et al., 2024; Gao et al., 2024; Li et al., 2024; Gou et al., 2023). These models expand the conventional single feed-forward network (FFN) architecture and its corresponding representation space into multiple subspaces, thereby effectively emulating the behavior of MoE architectures.

Prominent examples include MoLA (Gao et al., 2024), LoRAMoE (Dou et al., 2024), and MixLoRA (Li et al., 2024), all of which have demonstrated state-of-the-art performance compared to other PEFT-based methods across various benchmarks. Consequently, we integrate these three distinct LoRA MoE methodologies into our GRAPHMOE framework. The primary differences between these models are as follows: LoRAMoE incorporates vanilla attention LoRA layers and MLP plugged with LoRA-MoE layers; MoLA integrates plugged LoRA-MoE layers in both attention and MLP layers; and MixLoRA combines fused LoRA-MoE modules in attention and MLP layers. Further analysis of these LoRA-MoE implementations can be found in Li et al. (2024).

In this study, we augment MoE architectures by enhancing their reasoning depth and demonstrate the efficacy of our novel architecture when integrated with PEFT-based LLM baselines. For a detailed review of related works, including general PEFT, Transformer architectures, and standard MoE, please refer to the Appendix A.

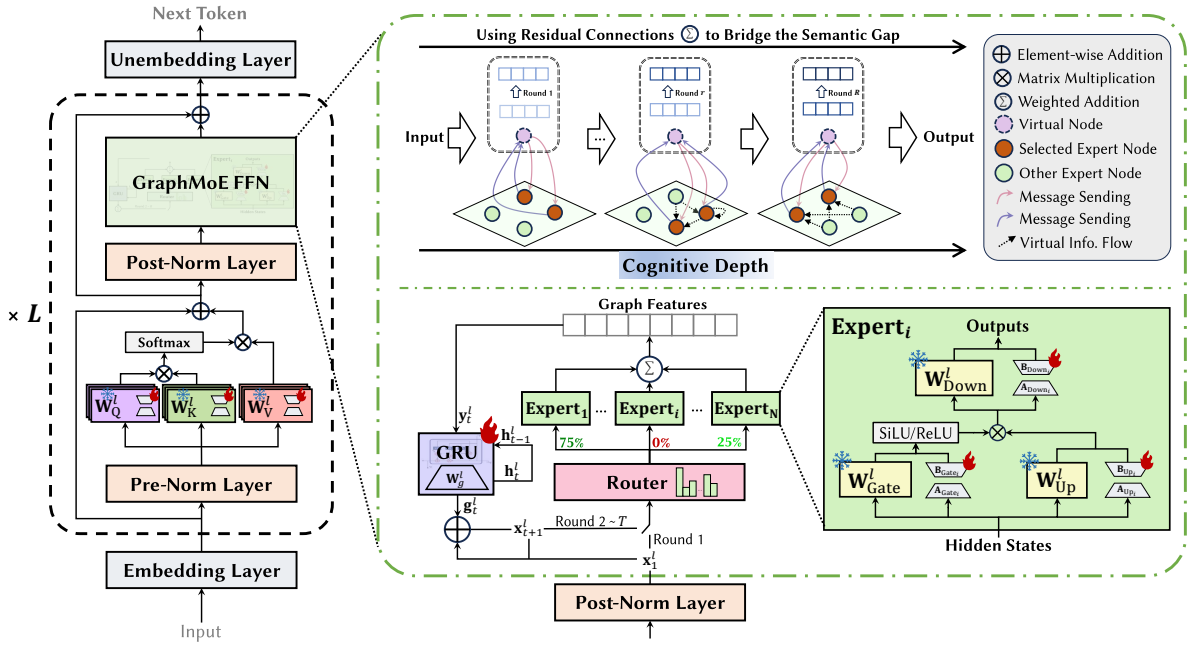


Figure 2: Overview of GRAPHMOE architecture. In this figure, the original Feed-Forward Network (FFN) layer in each transformer block is modified.

3 Method

We propose GRAPHMOE, a pseudo-graph-based MoE framework that connects experts as graph nodes to enable iterative information exchange, inspired by the modular reasoning patterns of the human brain. The overall approach includes three components: transforming existing LoRA-based MoE architectures, constructing a pseudo reasoning graph among experts, and training the model with a recurrent routing mechanism. As shown in Figure 2, the self-rethinking mechanism operates within the GraphMoE Feed-Forward Network (FFN) layer, which replaces the original FFN layer found in standard transformer-based LLMs. While MixLoRA is used as the base example, the framework is compatible with MoLA and LoRAMoE. Across all variants, a unified self-rethinking module based on GRU enables reasoning across expert layers.

3.1 MoE Transformation

Incorporating multiple expert models into LoRA layers, inspired by the work of Li et al. (2024) and Dou et al. (2024), the standard Feed-Forward Networks (FFNs) can be converted into MoE structures comprising n experts, denoted as $\{E_i\}_{i=1}^{n}$. As illustrated in the bottom-right corner of Figure 2, for the ℓ-th layer of the LLM ($1 \leq \ell \leq L$), we freeze the original FFN parameters and train three pairs of low-rank matrices for each expert: $\mathbf{A}^{\ell}_{\text{Down}_i} \in \mathbb{R}^{r \times d_1}$, $\mathbf{B}^{\ell}_{\text{Down}_i} \in \mathbb{R}^{d_2 \times r}$, $\mathbf{A}^{\ell}_{\text{Gate}_i}$, $\mathbf{B}^{\ell}_{\text{Gate}_i}$, $\mathbf{A}^{\ell}_{\text{Up}_i}$, and $\mathbf{B}^{\ell}_{\text{Up}_i}$. Here, i represents the i-th expert, and d_1, d_2 are the dimensions of the corresponding weight matrices $\mathbf{W}^{\ell}_{\text{Down}_i} \in \mathbb{R}^{d_2 \times d_1}$, with other dimensions defined similarly. Each expert $E_i(\cdot)$ is constructed as a fusion of the original FFN and trainable low-rank matrices, as shown in Equations 1 and 2, where $\sigma(\cdot)$ denotes activation functions (e.g., SiLU, ReLU), and $\odot$ denotes element-wise multiplication.

$$E^{\ell}_i(\mathbf{x}) = \tilde{\mathbf{W}}^{\ell}_{\text{Down}_i}\left(\sigma(\tilde{\mathbf{W}}^{\ell}_{\text{Gate}_i}\mathbf{x}) \odot (\tilde{\mathbf{W}}^{\ell}_{\text{Up}_i}\mathbf{x})\right) \quad (1)$$

$$\begin{aligned}
\tilde{\mathbf{W}}^{\ell}_{\text{Gate}_i} &= \mathbf{W}^{\ell}_{\text{Gate}_i} + \mathbf{B}^{\ell}_{\text{Gate}_i} \cdot \mathbf{A}^{\ell}_{\text{Gate}_i} \\
\tilde{\mathbf{W}}^{\ell}_{\text{Down}_i} &= \mathbf{W}^{\ell}_{\text{Down}_i} + \mathbf{B}^{\ell}_{\text{Down}_i} \cdot \mathbf{A}^{\ell}_{\text{Down}_i} \\
\tilde{\mathbf{W}}^{\ell}_{\text{Up}_i} &= \mathbf{W}^{\ell}_{\text{Up}_i} + \mathbf{B}^{\ell}_{\text{Up}_i} \cdot \mathbf{A}^{\ell}_{\text{Up}_i}
\end{aligned} \quad (2)$$

Additionally, we introduce a trainable linear layer $\mathbf{R}^{\ell} \in \mathbb{R}^{n \times d}$, referred to as the Router, where n denotes the total number of experts, and d represents the dimension of the hidden state. The Router determines the importance of each expert during forward propagation by computing a relevance score for every expert. Based on the Router's output, we select the top k experts and normalize their corresponding weights to ensure effective routing, as described in Equations 3 and 4. In subsequent experiments, unless stated otherwise, our setup activates 2 out of 8 experts, thus setting k to 2.

$$\hat{\mathbf{s}}^{\ell} = \text{Softmax}\left(\mathbf{R}^{\ell} \cdot \mathbf{x}\right) \quad (3)$$

$$\mathbf{s}^{\ell} = \text{Top-}k\left(\hat{\mathbf{s}}^{\ell}\right) = \begin{cases} \hat{\mathbf{s}}^{\ell}[i], & \text{if } \hat{\mathbf{s}}^{\ell}[i] \text{ is in the top } k, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Finally, as shown in Equation 5, the output of the MoE module is computed as a weighted sum of the selected experts. This result replaces the original output of the vanilla FFN module.

$$\mathbf{y}^{\ell} = MoE(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{s}^{\ell}[i] \cdot E_i^{\ell}(\mathbf{x}) \tag{5}$$

3.2 Pseudo Reasoning Graph Construction

The development of the pseudo reasoning graph represents a pivotal step in implementing our proposed self-rethinking mechanism, which is designed to enhance the internal consistency of LLMs (Liang et al., 2024). The pseudocode outlining the inference process of GRAPHMOE is detailed in Appendix B. For clarity and conciseness, the terms Gate, Up, and Down in the subscripts are abbreviated as G, U, and D, respectively.

Initially, the pseudo graph is constructed as illustrated in the top right corner of Figure 2. Within this framework, expert models are represented as graph nodes, each functioning according to the procedure described in §3.1. A distinctive virtual node is simultaneously established alongside these expert model graph nodes. Unlike other nodes, the virtual node is characterized by a graph feature vector and a GRU module. This node serves as a recurrent router, acting as a conduit for expert node connectivity and a layer for forward feature aggregation.

In contrast to conventional knowledge graphs, there is no prior knowledge that defines the edges between nodes. Instead, the edges connecting graph nodes are dynamically generated at each reasoning round, denoted as t, with the virtual node serving as the intermediary, as indicated by the black dashed arrows in Figure 2. This process facilitates the dynamical activation of temporal edges between nodes, enabling feature transformation across nodes and the newly created edges. Consequently, expert nodes at each reasoning round can communicate and collaborate to address the given task effectively.

Therefore, the core of this method lies in implementing the functionality of the virtual node. In this work, we employ a Low-Rank GRU to serve as the virtual node. Specifically, we reduce the GRU's hidden size to $\bar{d} \ll d$ and incorporate a low-rank linear layer to compute its output $\mathbf{g}_t^{\ell}$. The virtual node integrates features from previous reasoning round and processes the expert results $\mathbf{y}^{\ell}$, subsequently selecting k experts for the next round.

The reasoning (self-thinking) process involves a total of T rounds of MoE computations. The first round is processed as described in Equations 1-5. After obtaining the representation $\mathbf{y}_t^{\ell}$ from the MoE, it is fed into the Low-Rank GRU along with the hidden state $\mathbf{h}_{t-1}^{\ell}$ from the previous timestep (here, "round" and "timestep" are equivalent since there is only one step per round). A residual connection is applied by adding the result to the input of the previous round, $\mathbf{x}_t^{\ell}$, to produce the input for the next round, $\mathbf{x}_{t+1}^{\ell}$. This iterative process extracts meta-information for subsequent reasoning rounds, as shown in Equations 6-7.

$$\begin{aligned} \mathbf{z}_t^{\ell} &= \sigma\left(\mathbf{W}_z^{\ell} \cdot \left[\mathbf{h}_{t-1}^{\ell}, \mathbf{y}_t^{\ell}\right]\right), \mathbf{W}_z^{\ell} \in \mathbb{R}^{\bar{d} \times (\bar{d}+d)} \\ \mathbf{r}_t^{\ell} &= \sigma\left(\mathbf{W}_r^{\ell} \cdot \left[\mathbf{h}_{t-1}^{\ell}, \mathbf{y}_t^{\ell}\right]\right), \mathbf{W}_r^{\ell} \in \mathbb{R}^{\bar{d} \times (\bar{d}+d)} \\ \hat{\mathbf{h}}_t^{\ell} &= \sigma\left(\mathbf{W}_o^{\ell} \cdot \left[\mathbf{r}_t^{\ell} \odot \mathbf{h}_{t-1}^{\ell}, \mathbf{y}_t^{\ell}\right] + \mathbf{b}_o^{\ell}\right) \\ \mathbf{h}_t^{\ell} &= \left(\mathbf{1} - \mathbf{z}_t^{\ell}\right) \odot \mathbf{h}_{t-1}^{\ell} + \mathbf{z}_t^{\ell} \odot \hat{\mathbf{h}}_t^{\ell} \end{aligned} \tag{6}$$

$$\begin{aligned} \mathbf{g}_t^{\ell} &= \mathbf{W}_g^{\ell} \cdot \mathbf{h}_t^{\ell}, \mathbf{W}_g^{\ell} \in \mathbb{R}^{d \times \bar{d}} \\ \mathbf{x}_{t+1}^{\ell} &= \mathbf{x}_t^{\ell} + \mathbf{g}_t^{\ell} \end{aligned} \tag{7}$$

In a nutshell, from a model architecture standpoint, the GRU and the Router together form the aforementioned "virtual node". Specifically, the GRU functions to aggregate information from multiple experts E_i, whereas the Router is responsible for determining which experts should receive the integrated information for further inference, as depicted in Figure 1 in Appendix A.

3.3 GRAPHMOE Training

Trainable Components. As shown in Figure 2, the modules marked with a flame icon represent the trainable components in GRAPHMOE. Specifically, we need to train $3 \cdot L \cdot n$ pairs of low-rank adapters applied to the MoE modules, as well as a single linear layer serving as the Router. Additionally, in each decoder layer, the Low-Rank GRU module requires training 4 linear layers: $\mathbf{W}_z^{\ell}$, $\mathbf{W}_r^{\ell}$, $\mathbf{W}_o^{\ell}$, and $\mathbf{W}_g^{\ell}$.

Inspired by Li et al. (2024), we also enhance the attention module by adding 4 pairs of trainable low-rank matrices to each decoder layer. These matrices are applied to the attention components $\mathbf{Q}_h^{\ell}$, $\mathbf{K}_h^{\ell}$, $\mathbf{V}_h^{\ell} \in \mathbb{R}^{d \times \frac{d}{H}}$, and $\mathbf{O}_h^{\ell} \in \mathbb{R}^{\frac{d}{H} \times d}$, where h and H represent the h-th attention head and the total number of heads,

respectively. This enhances the adaptability of certain attention heads to specific tasks (Zheng et al., 2024).

In summary, as shown in Table 1, GRAPHMOE training involves a total of $(3n+4)\cdot L$ pairs of low-rank adapters and $4\cdot L$ linear layers.

Component	Attention Head	MoE	GRU	Total
Low-rank adapters	$4\cdot L$	$3\cdot L\cdot n$	0	$(3n+4)\cdot L$
Linear layers	0	0	$4\cdot L$	$4\cdot L$

Table 1: Summary of components to be trained.

Loss Design. Similar to standard continual learning (Wu et al., 2024c) and supervised fine-tuning (Patil and Gudivada, 2024), the primary objective during training is to minimize the cross-entropy loss $\mathcal{L}_{CE}$ between the predicted token logits and the target tokens.

However, the Router is a unique and critical component of the MoE architecture, as it determines which experts are activated during inference. Prior research has shown that unconstrained Router training can lead to over-reliance on a few specific experts (Zoph et al., 2022; Fedus et al., 2022), resulting in poor load balancing. Inspired by the work of Shen et al. (2024) and Muennighoff et al. (2024), we introduce a **Load Balancing Loss** $\mathcal{L}_{LB}$ (as defined in Equation 8) as an auxiliary objective. This loss penalizes the Router for disproportionately selecting the same experts across the sequence set $\mathcal{M}$, which consists of several sequences and the corresponding next tokens to be predicted.

$$\begin{aligned} \mathcal{L}_{LB} &= n\cdot\sum_{i=1}^{n} f_i\cdot p_i, \\ \text{where } f_i &= \frac{1}{|\mathcal{M}|}\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{M}} \mathbb{1}\{\mathbf{s}_{\mathbf{x}}[i]>0\}, \\ p_i &= \frac{1}{|\mathcal{M}|}\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{M}} \mathbf{s}_{\mathbf{x}}[i] \end{aligned} \tag{8}$$

Setting a hyperparameter λ, the total loss can be expressed as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{CE} + \lambda\cdot\mathcal{L}_{LB} \tag{9}$$

4 Experiment

4.1 Experimental Setup

Datasets. We evaluate our method on a diverse range of benchmarks: ARC (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), and SocialIQA (Sap et al., 2019) for QA tasks; BoolQ (Clark et al., 2019) for classification tasks; Hellaswag (Zellers et al., 2019) for science completion; and Winogrande (Sakaguchi et al., 2021) for fill-in-the-blank tasks.

Baselines. We adopt LLaMA-3-8B as the backbone for all models. To benchmark GRAPHMOE, we compare it with standard LoRA (Hu et al., 2021a) and three recent state-of-the-art LoRA+MoE methods: MoLA (Gao et al., 2024), LoRAMoE (Dou et al., 2024), and MixLoRA (Li et al., 2024). As Weight-Decomposed LoRA (DoRA) (Liu et al., 2024) has shown to improve LoRA's performance, we also implement DoRA-enhanced variants of each baseline.

Evaluation Metrics. We evaluate all methods using accuracy across all datasets. For Multiple-Choice QA tasks, we adopt the first-token evaluation strategy, which measures the probability of each option letter (Yu et al., 2024). To analyze expert utilization, we track the selection frequency of each expert during inference and compute their selection ratios. Additionally, we report several indicators: GRU hidden size scale ($\bar{d}/d$), parameter ratio (Param), and normalized inference time (Infer Time). GRU hidden size scale is defined as the ratio of GRU hidden size to model dimension; Param denotes the proportion of trainable parameters; Infer Time is normalized by dividing each model's inference time by the minimum across all models.

Implementation Details. All methods use LLaMA-3-8B[1] as the backbone. Training settings follow Li et al. (2024). The key hyperparameters for all methods are as follows: the cutoff length is set to 512, the learning rate is 2e-4, and the optimizer used is AdamW. We use a batch size of 16 with 8 accumulation steps, and the dropout rate is 0.05. Training is conducted for 2 epochs. For LoRA and DoRA methods, the rank r is set to 80, and the LoRA alpha α is set to 160. For MixLoRA, MixDoRA and GRAPHMOE, the LoRA rank and α is set to 16 and 32, respectively. Each MoE layer consists of 8 experts, with top-2 experts activated during inference. For GRAPHMOE, (1) reasoning round $T=3$ (the first round disables GRU-based feature transmission); (2) a 1-layer GRU with a hidden size scaled to 0.1×4096; (3) a load-balance loss weight $\lambda=0.01$. All experiments are conducted on a single A800 GPU using BF16 precision[2].

4.2 Experimental Result

Effect of GRAPHMOE. As shown in Table 2, integrating GRAPHMOE, a pseudo graph-based

[1] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

[2] Additional implementation details are provided in Appendix C

Method	ARC-E	ARC-C	BoolQ	OBQA	PIQA	SIQA	HellaS	WinoG	AVG.
MoLA	86.4	77.9	74.0	84.4	86.7	76.4	93.9	83.3	82.9
GRAPHMOE (MoLA)	89.7	81.0	75.8	87.6	87.8	79.7	95.5	83.2	**85.0**
Δ	3.3	3.1	1.8	3.2	1.1	3.3	1.6	-0.1	2.1
LoRAMoE	87.8	79.5	72.4	85.0	87.1	74.8	94.8	83.4	83.1
GRAPHMOE (LoRAMoE)	87.7	79.9	75.9	88.8	87.8	79.4	94.8	83.4	**84.7**
Δ	-0.1	0.4	3.5	3.8	0.7	4.6	0.0	0.0	1.6
MixLoRA	86.9	77.0	74.0	84.4	86.0	75.5	93.7	83.3	82.6
GRAPHMOE (MixLoRA)	89.2	80.9	75.8	89.6	87.8	77.0	95.4	83.2	**84.9**
Δ	2.3	3.9	1.8	5.2	1.8	1.5	1.7	-0.1	2.3

Table 2: Comparison of the performance between baseline LoRA+MoE models and GRAPHMOE (LoRA+MoE) models. GRAPHMOE (·) indicates that the conventional MoE module in the baseline has been substituted with the self-rethinking mechanism proposed in § 3.2. Scores highlighted in **bold** denote superior performance for each respective method. Green and red indicate increase and decrease in performance following the application of GRAPHMOE respectively.

Method	Param	ARC-E	ARC-C	BoolQ	OBQA	PIQA	SIQA	HellaS	WinoG	AVG.
Base (ICL-3)	0%	90.6	76.7	45.9	73.8	61.6	71.5	64.1	59.2	65.6
Base (ICL-3, Self-cons-5)	0%	91.1	79.4	50.2	73.7	57.0	70.3	66.9	62.1	66.3
Base (ICL-3, Self-cons-10)	0%	91.6	78.1	42.8	75.3	59.8	71.3	69.9	61.4	66.2
Base (ICL-3, Self-cons-15)	0%	91.9	78.0	45.6	72.0	67.9	71.7	63.3	63.2	66.4
LoRA*	2.6%	89.0	75.7	67.2	85.0	80.7	78.3	74.2	75.3	78.2
DoRA*	2.6%	88.1	76.4	61.7	80.6	82.3	76.2	78.8	83.7	78.5
MixLoRA*	3.0%	86.5	79.9	75.0	84.8	87.6	78.8	93.3	82.1	83.5
MixDoRA*	3.0%	87.7	78.9	76.8	86.9	83.4	80.1	94.6	84.2	84.1
LoRA	2.6%	86.2	76.7	70.2	78.2	82.9	74.2	80.1	50.7	74.9
DoRA	2.6%	87.6	76.8	69.3	80.8	84.2	76.8	85.3	50.4	76.4
MoLA	2.7%	86.4	77.9	74.0	84.4	86.7	76.4	93.9	83.3	82.9
MoDA	2.7%	86.4	78.0	74.0	84.8	87.4	76.4	**95.5**	**84.1**	83.3
LoRAMoE	3.2%	87.8	79.5	72.4	85.0	87.1	74.8	94.8	83.4	83.1
DoRAMoE	3.2%	87.9	80.1	74.6	85.8	86.3	74.5	94.0	83.1	83.3
MixLoRA	3.0%	86.9	77.0	74.0	84.4	86.0	75.5	93.7	83.3	82.6
MixDoRA	3.0%	86.7	76.7	75.5	84.8	85.2	76.9	93.0	83.1	82.7
GRAPHMOE (MixLoRA)	5.9%	89.2	**80.9**	75.8	**89.6**	87.8	77.0	95.4	83.2	84.9
GRAPHMOE (MixDoRA)	5.9%	**90.3**	80.6	**75.9**	88.2	**88.8**	**79.4**	95.3	83.7	**85.3**

Table 3: Performance of LoRA-based SFT methods across benchmarks. ICL-n" indicates n in-context examples; Self-cons-5" denotes 5 samples for self-consistency. Param" is the number of trainable parameters. Results marked with * are from Li et al. (2024) and use full precision (FP32); all others use BF16. DoRA-based variants are denoted by replacing L" with "D" in model names.

MoE with recurrent routing, consistently improves performance over all SOTA LoRA+MoE baselines. This demonstrates GRAPHMOE's ability to enhance expert collaboration by simply replacing the original MoE module, enabling more effective feature exchange among experts. Performance gains vary across base models due to differences in how LoRA and MoE are combined. As noted by Li et al. (2024), MoLA and LoRAMoE use a "plugged-in" design, injecting LoRA modules into attention or MLP outputs, while MixLoRA adopts a "fused" strategy that embeds LoRA experts directly into the network layers—yielding stronger integration and performance boost when enhanced by GRAPHMOE. Across all tasks, GRAPHMOE consistently improves or at least maintains accuracy. Notably, on SocialIQA, applying GRAPHMOE to LoRAMoE improves accuracy from 74.8 to 79.4 (+4.6, over 6%). These observations underscores the effectiveness of the GRAPHMOE approach within this architectural framework.

Main Comparison Results. The experimental results are presented in Table 3. We compare the performance of four base models (LLaMA-3-8B) incorporating in-context learning (ICL-n) (Dong et al., 2024) and self-consistency (Self-cons-n) (Wang et al., 2023). While increasing the number of samples for self-consistency to 5, 10, and 15 shows minimal variation in performance (scores of 66.3, 66.2, and 66.4, respectively), these models still fall short compared to those using LoRA-based SFT methods. Due to computational constraints, we use BF16 precision for training and inference. However, we include full

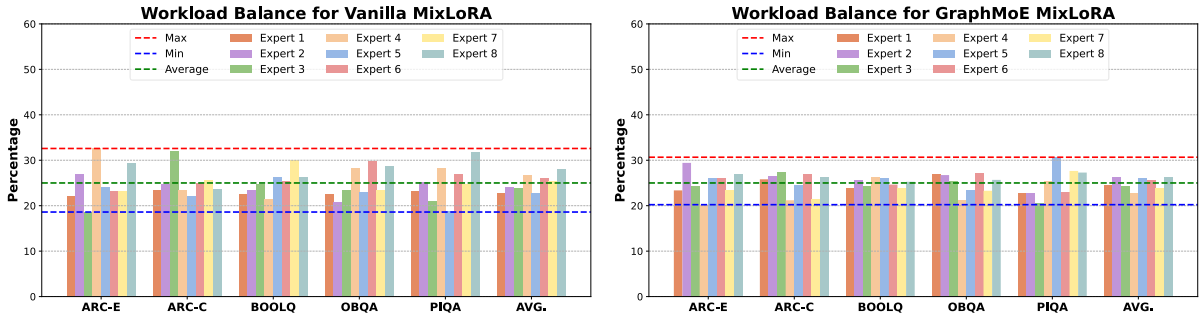
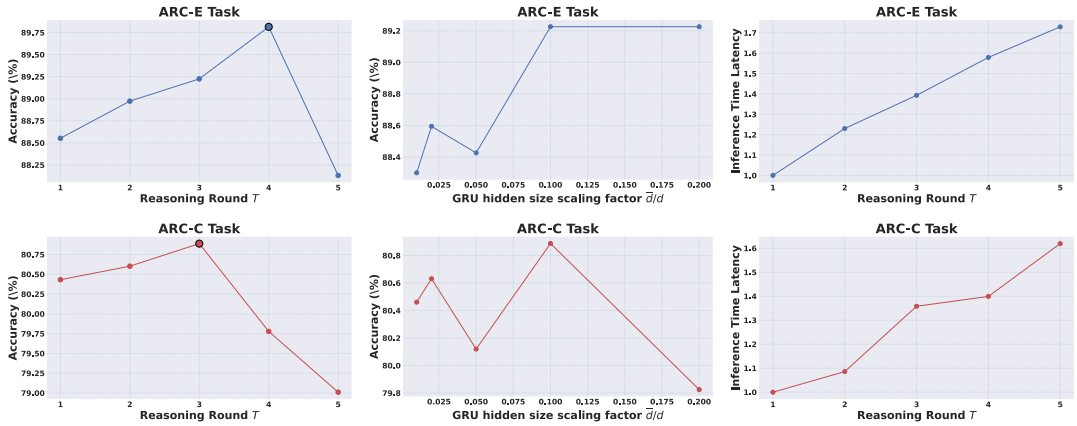


Figure 3: Normalized expert workloads across routing steps. Red and blue dashed lines mark the max and min workloads; green indicates the average. The workload std is 0.0313 for MixLoRA and 0.0215 for GRAPHMOE.



(a) Accuracy vs. Reasoning Round (b) Accuracy vs. GRU hidden size scaling factor (c) Infer. Time Latency vs. Reasoning Round

Figure 4: Sensitivity analysis of GRAPHMOE hyperparameters. Subfigure (c) shows how inference time scales with reasoning round (T), where the first round's time is set as the unit. "Infer. Time" denotes the relative time per round.

precision MixLoRA results for comparison. Despite a significant performance drop in all models when using reduced precision, our GRAPHMOE models with MixLoRA and MixDoRA still outperform full precision models, demonstrating the efficacy of our architecture. Moreover, MixLoRA does not represent the optimal base model with the LoRA+MoE framework, as MoLA and LoRAMoE have better accuracy scores. This indicates that the performance gains achieved by GRAPHMOE surpass those of existing methodologies applied within the LoRA+MoE framework. The self-rethinking mechanism enhances the MoE approach, evident in both LoRA and DoRA settings. This suggests that the improvements carried by the PRG are not merely a result of simple adjustments on additional trainable parameters, but rather arise from the effective collaboration among diverse expert models facilitated by our approach.

Analysis of Workload Balance. The main difference between MixLoRA and GRAPHMOE (MixLoRA) is the self-rethinking mechanism enabled by the recurrent router. We analyze the impact of expert model selection at each routing stage. Figure 3 compares the workload distribution across expert models in MixLoRA and GRAPHMOE (MixLoRA) for various tasks. Results show that GRAPHMOE achieves more balanced expert selection, with standard deviations of 0.0313 for MixLoRA and 0.0249 for GRAPHMOE (MixLoRA), a 20% reduction. This indicates that the reasoning process in GRAPHMOE activates a broader range of experts, enhancing MoE's representation learning. In each reasoning step, a distinct set of expert models is more likely to be activated to incrementally construct the representations within the MoE. This approach allows more expert models to operate in their optimal conditions, sequentially addressing complex problems. It is as if a chain-of-thoughts (CoT) is implicitly formed, using different expert models to fit features at different stages.

Sensitivity Analysis and Overhead. We further conduct sensitivity analysis on ARC-E and ARC-C to provide insights into the efficacy of our self-rethinking mechanism. In subfigure (a) of Figure 4, increasing the Reasoning Round T improves accuracy for both

Method	Param	ARC-E	ARC-C	BoolQ	OBQA	PIQA	SIQA	HellaS	WinoG	AVG
LoRA	1.9%	87.5	78.9	69.8	84.4	84.4	77.1	88.4	49.1	77.5
LoRA	2.6%	86.2	76.7	70.2	78.2	82.9	74.2	80.1	50.7	74.9
LoRA	5.9%	84.7	78.6	63.6	82.2	82.1	32.9	84.1	50.4	69.8
LoRA	7.9%	83.9	73.1	62.2	84.4	82.5	74.6	92.7	49.5	75.4
LoRA	9.9%	85.9	73.3	70.0	82.0	85.7	32.1	89.1	49.6	71.0
MixLoRA	3.0%	86.9	77.0	74.0	84.4	86.0	75.5	93.7	83.3	82.6
MixLoRA	6.0%	87.3	79.0	74.9	88.4	88.1	76.7	93.6	82.5	83.8
MixLoRA	9.0%	86.7	77.0	71.7	86.2	80.5	74.0	89.5	80.4	80.8
GRAPHMOE	5.9%	**89.2**	**80.9**	**75.8**	**89.6**	**87.8**	**77.0**	**95.4**	**83.2**	**84.9**

Table 4: Performance Comparison of Parameter Variants. In the table, GRAPHMOE refers to the GRAPHMOE (MixLoRA) model.

Method	Param	ARC-E	ARC-C	BoolQ	OBQA	PIQA	SIQA	HellaS	WinoG	AVG
MixLoRA†	6.0%	87.4	79.2	73.9	85.2	85.3	72.9	95.0	80.4	82.4
MixLoRA	6.0%	87.3	79.0	74.9	88.4	88.1	76.7	93.6	82.5	**83.8**
Δ	-	-0.1	-0.2	1	3.2	2.8	3.8	-1.4	2.1	1.4
GRAPHMOE†	5.9%	88.1	79.0	74.1	87.8	85.6	74.3	93.2	85.1	83.4
GRAPHMOE	5.9%	89.2	80.9	75.8	89.6	87.8	77.0	95.4	83.2	**84.9**
Δ	-	1.1	1.9	1.7	1.8	2.2	2.7	2.2	-1.9	1.5

Table 5: Performance Comparison of GRAPHMOE (MixLoRA) with different activated experts. GRAPHMOE† indicates the GRAPHMOE (MixLoRA) variant with 4 out of 8 experts activated, whereas the other LoRA MoE models in above experiments have 2 out of 8 experts activated.

tasks, demonstrating the self-rethinking mechanism's effectiveness in enhancing MoE. However, accuracy drops significantly beyond a certain threshold ($T=4$ for ARC-E and $T=3$ for ARC-C), likely due to overfitting from "overthinking" the problem. Subfigure (b) of Figure 4 further illustrates that increasing the GRU hidden size does not always improve performance, highlighting the importance of the self-rethinking mechanism over the added GRU parameters. Finally, subfigure (c) shows that, compared to conventional LoRA+MoE models (MixLoRA), which employ a single reasoning round, the increase in inference time for GRAPHMOE is only approximately 0.16 times for each additional reasoning round. These findings shows the promising impact GRAPHMOE on enhancing MoE's cognitive depth while maintaining acceptable computational overhead.

Analysis of PEFT Parameters. We analyze the effect of the parameter size on model performance. From Table 4, GRAPHMOE (MixLoRA) gains the best overall performance, which uses 5.9% of parameters yet achieves the highest average score of 84.9. In addition, it can be observed that simply increasing parameter size does not lead to proportional improvements in SFT performance. For instance, MixLoRA with 3.0% of parameters achieves the best average score of 82.6%, while increasing the parameter size to 9.0% causes fluctuations below that score. Similarly, in Figure 4(b), performance does not improve as the GRU hidden size increases.

Analysis of Active Expert Number. To further investigate MoE utilization, we tested the effect of activating additional experts in the MoE layers, as shown in Table 5. Surprisingly, increasing the number of active experts led to a performance decline. This highlights that simply adding parameters does not guarantee better reasoning performance. Instead, the effectiveness of the neural architectural, as demonstrated in Table 4, is a more critical factor.

5 Conclusion

In this work, we introduce GRAPHMOE, a novel approach that enhances the cognitive depth of language models through the integration of a self-rethinking mechanism in pseudo-graph Mixture-of-Experts (MoE) networks. Unlike traditional MoE models that operate independently, GRAPHMOE facilitates communication among expert nodes, allowing for iterative refinement and enhanced reasoning capabilities. Implemented using Low-Rank Adaptation techniques (LoRA), GRAPHMOE demonstrates significant performance improvements across various benchmark datasets, outperforming existing LoRA & LoRA+MoE baseline models and achieving state-of-the-art results.

Limitations

Despite the notable performance improvements achieved by the GRAPHMOE framework, there are several limitations to be acknowledged.

- While the GRAPHMOE architecture consistently leads to performance gains across diverse tasks and base models, the degree of improvement is variable and can depend on the interaction between the MoE and LoRA components. This variability underscores the need for comprehensive exploration into the integration modalities and parameter space to ensure maximum efficiency across different LoRA+MoE configurations.

- The workload distribution analysis indicates a marked improvement in workload balance with GRAPHMOE. However, the potential impact of workload imbalance on model performance over a broader range of tasks remains underexplored. Further investigation into balancing expert model selection and activation across diverse scenarios could unveil additional pathways for performance optimization.

- The sensitivity analysis indicated potential overfitting issues when increasing the reasoning rounds beyond a particular threshold, revealing the necessity for careful hyperparameter tuning to mitigate issues of over-complexity and model overthinking.

In summary, while the GRAPHMOE framework shows promise in enhancing cognitive depth and performance of MoE architectures, future research should address computational precision limits, explore broader integration strategies, and focus on hyperparameter optimization to further refine and substantiate these initial findings.

References

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

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. 2024. A survey on mixture of experts. *arXiv preprint arXiv:2407.06204*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

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

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, and 1 others. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Qingxiu Dong, Lei Li a@miscdong2024surveyincontextlearning, title=A Survey on In-context Learning, author=Qingxiu Dong and Lei Li and Damai Dai and Ce Zheng and Jingyuan Ma and Rui Li and Heming Xia and Jingjing Xu and Zhiyong Wu and Tianyu Liu and Baobao Chang and Xu Sun and Lei Li and Zhifang Sui, year=2024, eprint=2301.00234, archivePrefix=arXiv, primaryClass=cs.CL, url=https://arxiv.org/abs/2301.00234, nd Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. *Preprint*, arXiv:2301.00234.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945, Bangkok, Thailand. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Tianyu Fu, Haofeng Huang, Xuefei Ning, Genghan Zhang, Boju Chen, Tianqi Wu, Hongyi Wang, Zixiao Huang, Shiyao Li, Shengen Yan, and 1 others. 2024. Moa:

Mixture of sparse attention for automatic large language model compression. *arXiv preprint arXiv:2406.14909*.

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

Tomas Goldsack, Zhihao Zhang, Chen Tang, Carolina Scarton, and Chenghua Lin. 2023. Enhancing biomedical lay summarisation with external knowledge graphs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8016–8032.

Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T Kwok, and Yu Zhang. 2023. Mixture of cluster-conditional lora experts for vision-language instruction tuning. *arXiv preprint arXiv:2312.12379*.

Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, Deyi Xiong, and 1 others. 2023. Evaluating large language models: A comprehensive survey. *arXiv preprint arXiv:2310.19736*.

Xu Owen He. 2024. Mixture of a million experts. *arXiv preprint arXiv:2407.04153*.

S Hochreiter. 1997. Long short-term memory. *Neural Computation MIT-Press*.

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

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

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

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

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

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

Xun Liang, Shichao Song, Zifan Zheng, Hanyu Wang, Qingchen Yu, Xunkai Li, Rong-Hua Li, Yi Wang, Zhonghao Wang, Feiyu Xiong, and 1 others. 2024. Internal consistency and self-feedback in large language models: A survey. *arXiv preprint arXiv:2407.14507*.

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

Tyler Loakman, Chen Tang, and Chenghua Lin. 2023. TwistList: Resources and baselines for tongue twister generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

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

Bo Lv, Chen Tang, Yanan Zhang, Xin Liu, Ping Luo, and Yue Yu. 2024. URG: A unified ranking and generation method for ensembling language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 4421–4434, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, and 1 others. 2024. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*.

Mohammed Muqeeth, Haokun Liu, and Colin Raffel. 2023. Soft merging of experts with adaptive routing. *arXiv preprint arXiv:2306.03745*.

Rajvardhan Patil and Venkat Gudivada. 2024. A review of current trends, techniques, and challenges in large language models (llms). *Applied Sciences*, 14(5):2074.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, and 1 others. 2023. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*.

Subhojeet Pramanik, Esraa Elelimy, Marlos C Machado, and Adam White. 2023. Recurrent linear transformers. *arXiv preprint arXiv:2310.15719*.

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Yikang Shen, Zhen Guo, Tianle Cai, and Zengyi Qin. 2024. Jetmoe: Reaching llama2 performance with 0.1 m dollars. *arXiv preprint arXiv:2404.07413*.

Chen Tang. 2024. *Knowledge Enhanced Natural Language Generation*. Ph.D. thesis, University of Surrey.

Chen Tang, Hongbo Zhang, Tyler Loakman, Chenghua Lin, and Frank Guerin. 2023. Enhancing dialogue generation via dynamic graph knowledge aggregation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Shaohua Wu, Jiangang Luo, Xi Chen, Lingjun Li, Xudong Zhao, Tong Yu, Chao Wang, Yue Wang, Fei Wang, Weixu Qiao, and 1 others. 2024a. Yuan 2.0-m32: Mixture of experts with attention router. *arXiv preprint arXiv:2405.17976*.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. 2024b. Mixture-of-subspaces in low-rank adaptation. *arXiv preprint arXiv:2406.11909*.

Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. 2024c. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364*.

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. 2024. Improving transformers with dynamically composable multi-head attention. *arXiv preprint arXiv:2405.08553*.

Junbing Yan, Chengyu Wang, Taolin Zhang, Xiaofeng He, Jun Huang, and Wei Zhang. 2023. From complex to simple: Unraveling the cognitive tree for reasoning with small language models. *arXiv preprint arXiv:2311.06754*.

Bohao Yang, Chen Tang, Kun Zhao, Chenghao Xiao, and Chenghua Lin. 2024. Effective distillation of table-based reasoning ability from LLMs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5538–5550, Torino, Italia. ELRA and ICCL.

Qingchen Yu, Zifan Zheng, Shichao Song, Zhiyu Li, Feiyu Xiong, Bo Tang, and Ding Chen. 2024. xfinder: Robust and pinpoint answer extraction for large language models. *arXiv preprint arXiv:2405.11874*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Wei Hu Muhao Chen Jian Dai Wei Zhang Yuzhong Qu Zequn Sun, Chengming Wang. 2020. Knowledge graph alignment network with gated multi-hop neighborhood aggregation. In *AAAI*, pages 222–229.

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

Zhaoyang Zhang, Wenqi Shao, Yixiao Ge, Xiaogang Wang, Jinwei Gu, and Ping Luo. 2024. Cached transformers: Improving transformers with differentiable memory cachde. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 16935–16943.

Kun Zhao, Chenghao Xiao, Chen Tang, Bohao Yang, Kai Ye, Noura Al Moubayed, Liang Zhan, and Chenghua Lin. 2024a. X-ray made simple: Radiology report generation and evaluation with layman's terms. *arXiv preprint arXiv:2406.17911*.

Kun Zhao, Bohao Yang, Chenghua Lin, Wenge Rong, Aline Villavicencio, and Xiaohui Cui. 2023. Evaluating open-domain dialogues in latent space with next sentence prediction and mutual information. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 562–574.

Kun Zhao, Bohao Yang, Chen Tang, Chenghua Lin, and Liang Zhan. 2024b. Slide: A framework integrating small and large language models for open-domain dialogues evaluation. *arXiv preprint arXiv:2405.15924*.

Zifan Zheng, Yezhaohui Wang, Yuxin Huang, Shichao Song, Mingchuan Yang, Bo Tang, Feiyu Xiong, and Zhiyu Li. 2024. Attention heads of large language models: A survey. *arXiv preprint arXiv:2409.03752*.

Zexuan Zhong, Mengzhou Xia, Danqi Chen, and Mike Lewis. 2024. Lory: Fully differentiable mixture-of-experts for autoregressive language model pre-training. *arXiv preprint arXiv:2405.03133*.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*.

Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Tuo Zhao, and Jianfeng Gao. 2021. Taming sparsely activated transformer with stochastic experts. *arXiv preprint arXiv:2110.04260*.

A Related Work

A.1 Parameter-Efficient Fine-Tuning (PEFT)

LLMs have demonstrated remarkable improvements in general natural language understanding (NLU) and natural language generation (NLG) tasks (Guo et al., 2023; Chang et al., 2024; Tang, 2024). However, their performance often suffers from limited generalization and effectiveness in domain-specific applications. To address this, numerous studies have proposed Parameter-Efficient Fine-Tuning (PEFT) methods,

which optimize only a small subset of LLM parameters, significantly reducing computational overhead.

One of the most popular PEFT methods is Low-Rank Adaptation (LoRA) (Hu et al., 2021b), which introduces two low-rank matrices to each weight matrix $\mathbf{W}$ in LLMs. The product of these matrices represents the weight adjustment $\Delta\mathbf{W}$. Building upon LoRA, subsequent works have introduced more efficient variants, such as VeRa (Kopiczko et al., 2023), AdaLoRA (Zhang et al., 2023), DoRA (Liu et al., 2024), and MoSLoRA (Wu et al., 2024b). These methods aim to better capture task-specific features and integrate diverse feature subspaces effectively.

A.2 Transformer & Recurrent Models

The self-attention mechanism, as a fundamental algorithm within Transformers (Vaswani, 2017), facilitates parallel sequence processing and enhances model capacity through increased width. However, it lacks the temporal reasoning capabilities inherent in recurrent architectures, which contribute to model depth, such as Long Short-Term Memory (LSTM) networks (Hochreiter, 1997) and Gated Recurrent Units (GRUs) (Chung et al., 2014). To address this limitation, recent research has investigated integrating Transformer architectures with recurrent structures (Peng et al., 2023; Pramanik et al., 2023; Zhang et al., 2024; Tang, 2024). These efforts aim to blend recurrent neural network capabilities with the ability to capture long-distance dependencies. In this study, we likewise endeavor to incorporate a recurrent mechanism to model multiple expert systems. However, rather than focusing on capturing long-distance dependencies, our primary objective is to emulate the stepwise cognitive processes characteristic of human cognition. To achieve this, we employ GRUs to augment reasoning depth by aggregating hidden representations from attention features at each stage of the recurrent routing process. This integration is designed to enhance the model's proficiency in apprehending complex dependencies and thereby improve its overall reasoning capabilities.

A.3 Mixture of Experts (MoE)

The concept of Mixture of Experts (MoE) was first introduced by Jacobs et al. (1991), who proposed training multiple networks (experts) on different subsets of data and aggregating their outputs. Recently, as LLMs have become a focal point of research, MoE layers have been integrated into Transformer-based architectures. Specifically, researchers have replaced standard Feed-Forward Networks (FFNs) with sparse MoE layers, employing novel routing strategies (Zuo et al., 2021; Zhong et al., 2024; Wu et al., 2024a; Muqeeth et al., 2023; Fu et al., 2024) and advanced expert segmentation techniques (Dai et al., 2024; He, 2024; Jiang et al., 2024; Xiao et al., 2024).

Numerous studies have investigated the application of Parameter-Efficient Fine-Tuning (PEFT) methods to introduce additional trainable parameters for implementing pseudo MoE structures within LLMs (Dou et al., 2024; Luo et al., 2024; Gao et al., 2024; Li et al., 2024; Gou et al., 2023). These models expand the conventional single feed-forward network (FFN) architecture and its corresponding representation space into multiple subspaces, thereby effectively emulating the behavior of MoE architectures.

Prominent examples include MoLA (Gao et al., 2024), LoRAMoE (Dou et al., 2024), and MixLoRA (Li et al., 2024), all of which have demonstrated state-of-the-art performance compared to other PEFT-based methods across various benchmarks. Consequently, we integrate these three distinct LoRA MoE methodologies into our GRAPHMOE framework. The primary differences between these models are as follows: LoRAMoE incorporates vanilla attention LoRA layers and MLP plugged with LoRA-MoE layers; MoLA integrates plugged LoRA-MoE layers in both attention and MLP layers; and MixLoRA combines fused LoRA-MoE modules in attention and MLP layers. Further analysis of these LoRA-MoE implementations can be found in Li et al. (2024).

In this study, we augment MoE architectures by enhancing their reasoning depth and demonstrate the efficacy of our novel architecture when integrated with PEFT-based LLM baselines.

B Pseudo Reasoning Graph Construction Algorithm

To enhance clarity in illustrating the long processing pipeline, we summarize the entire methodology in Algorithm 1. The process involves T reasoning iterations, referred to as self-rethinking. Initially, features from the FFN layer of transformer blocks are derived via a LoRA MoE forward pass. Subsequently, these features are transmitted to the subsequent Low-Rank GRU module, which computes both the residual connection within the pseudo graph and the merged feature for the input of the succeeding iteration.

C Additional Implementation Details

Our hyperparameters strictly follow the configurations detailed in (Li et al., 2024) including the LoRA and DoRA methods and their three MoE derivatives, e.g.

Algorithm 1: GRAPHMOE at Layer ℓ

Input: $\mathbf{x}_1^\ell$: (B,P,D) after the Post-Norm.
Output: $\mathbf{y}_T^\ell$: (B,P,D) the hidden states.

/* B for Batch Num, P for Seq Len, D for Dim */
/* $\mathtt{D}'$ for Upper Dim, $\overline{\mathtt{D}}$ for GRU Hidden Dim */
for $t \leftarrow 1$ ***to*** T **do**
  /* MoE Forward */
  $\tilde{\mathbf{s}}_t^\ell$: (B,P,N) $\leftarrow \mathrm{Softmax}\left(\mathbf{R}^\ell * \mathbf{x}_t^\ell\right)$;
  $\mathbf{s}_t^\ell$: (B,P,N) $\leftarrow$ Top-$k\left(\tilde{\mathbf{s}}_1^\ell\right)$;
  for i *in* $\{$*selected experts* $E\}$ **do**
    $\tilde{\mathbf{W}}_{\mathrm{G}_i}^\ell$: ($\mathtt{D}'$,D) $\leftarrow \mathbf{W}_{\mathrm{G}_i}^\ell + \mathbf{B}_{\mathrm{G}_i}^\ell * \mathbf{A}_{\mathrm{G}_i}^\ell$;
    Similarly, we can get $\tilde{\mathbf{W}}_{\mathrm{U}_i}^\ell$ and $\tilde{\mathbf{W}}_{\mathrm{D}_i}^\ell$;
    $\hat{\mathbf{c}}_i^{\mathrm{G}}$: (B,P,$\mathtt{D}'$) $\leftarrow \tilde{\mathbf{W}}_{\mathrm{G}_i}^\ell * \mathbf{x}_t^\ell$;
    $\hat{\mathbf{c}}_i^{\mathrm{U}}$: (B,P,$\mathtt{D}'$) $\leftarrow \tilde{\mathbf{W}}_{\mathrm{U}_i}^\ell * \mathbf{x}_t^\ell$;
    $\hat{\mathbf{c}}_i$: (B,P,$\mathtt{D}'$) $\leftarrow \sigma\left(\hat{\mathbf{c}}_i^{\mathrm{G}}\right) \odot \hat{\mathbf{c}}_i^{\mathrm{U}}$;
    $\mathbf{c}_i$: (B,P,D) $\leftarrow \tilde{\mathbf{W}}_{\mathrm{D}_i}^\ell * \hat{\mathbf{c}}_i$;
    $\mathbf{y}_t^\ell$: (B,P,D) $\leftarrow \mathbf{y}_t^\ell + \mathbf{s}_t^\ell[:,i] \odot \mathbf{c}_i$;
  /* Low-Rank GRU */
  if $t \neq T$ **then**
    $\mathbf{z}_t^\ell$: (B,P,$\overline{\mathtt{D}}$) $\leftarrow \sigma\left(\mathbf{W}_z^\ell * \left[\mathbf{h}_{t-1}^\ell, \mathbf{y}_t^\ell\right]\right)$;
    $\mathbf{r}_t^\ell$: (B,P,$\overline{\mathtt{D}}$) $\leftarrow \sigma\left(\mathbf{W}_r^\ell * \left[\mathbf{h}_{t-1}^\ell, \mathbf{y}_t^\ell\right]\right)$;
    $\hat{\mathbf{h}}_t^\ell$: (B,P,$\overline{\mathtt{D}}$+D) $\leftarrow \left[\mathbf{r}_t^\ell \odot \mathbf{h}_{t-1}^\ell, \mathbf{x}_t^\ell\right]$;
    $\mathbf{h}_t^\ell$: (B,P,$\overline{\mathtt{D}}$) $\leftarrow \sigma\left(\mathbf{W}_o^\ell * \hat{\mathbf{h}}_t^\ell + \mathbf{b}_o^\ell\right)$;
    $\mathbf{h}_t^\ell \leftarrow \left(\mathbf{1} - \mathbf{z}_t^\ell\right) \odot \mathbf{h}_{t-1}^\ell + \mathbf{z}_t^\ell \odot \mathbf{h}_t^\ell$;
    $\mathbf{g}_t^\ell$: (B,P,D) $\leftarrow \mathbf{W}_g^\ell \cdot \mathbf{h}_t^\ell$;
    /* Residual Connection */
    $\mathbf{x}_{t+1}^\ell$: (B,P,D) $\leftarrow \mathbf{x}_t^\ell + \mathbf{g}_t^\ell$
return $\mathbf{y}_T^\ell$: (B,P,D);

MixLoRA or MixDoRA. We summarize the settings for LoRA/DoRA and their derivatives of MoE-based methods, as shown in Table 6. During evaluation, the batch size is set to 8.

Both techniques utilized a cutoff length of 512, a learning rate of $2e-4$, the AdamW optimizer, a batch size of 16, accumulation steps of 8, a dropout rate of 0.05, and were trained over 2 epochs. In LoRA/DoRA, the methods were applied to the Q, K, V, O, Up, Down, and Gate parameters with a LoRA rank r of 80 and a LoRA alpha α of 160. Conversely, MixLoRA/MixDoRA targeted the same parameters with a LoRA rank r of 16 and a LoRA alpha α of 32, incorporating 8 experts with a Top-k value of 2.

The experiments described in this paper required a single A800 GPU for at least one month. Including debugging and downtime, the entire process extended to around two months. The base model, LLaMA-8B,

Setting	LoRA/DoRA	MixLoRA/MixDoRA	GRAPHMOE
Cutoff Length	512		
Learning Rate	2e-4		
Optimizer	AdamW		
Batch Size	16		
Accumulation Steps	8		
Dropout Rate	0.05		
Training Epochs	2		
LoRA Rank (r)	80	16	
LoRA Alpha (α)	160	32	
Expert Number	-	8	
Top-k	-	2	
Targeted Parameters	$\mathbf{Q}_h^\ell$, $\mathbf{K}_h^\ell$, $\mathbf{V}_h^\ell$, $\mathbf{O}_h^\ell$, $\mathbf{W}_{\mathrm{Gate}_i}^\ell$, $\mathbf{W}_{\mathrm{Up}_i}^\ell$, $\mathbf{W}_{\mathrm{Down}_i}^\ell$		$\mathbf{Q}_h^\ell$, $\mathbf{K}_h^\ell$, $\mathbf{V}_h^\ell$, $\mathbf{O}_h^\ell$, $\mathbf{W}_{\mathrm{Gate}_i}^\ell$, $\mathbf{W}_{\mathrm{Up}_i}^\ell$, $\mathbf{W}_{\mathrm{Down}_i}^\ell$, $\mathbf{W}_z^\ell$, $\mathbf{W}_r^\ell$, $\mathbf{W}_o^\ell$, $\mathbf{W}_g^\ell$

Table 6: Training Settings for baselines and GRAPHMOE.

was obtained from the Huggingface repository "meta-llama/Meta-Llama-3-8B-Instruct"[3].

For experiments utilizing the GRAPHMOE framework, the key hyperparameters were configured as follows: (1) The number of GRU gate iterations within the recurrent routing—referred to as the reasoning (or self-rethinking) round T—was set to a default of 3. In the initial round, expert outputs are obtained without feature transmission via the GRU gate. (2) The GRU network consists of a single layer. (3) The GRU hidden size was scaled to 0.1×4096[4], where 4096 corresponds to the model dimension of LLaMA-3-8B. (4) The hyperparameter λ for the auxiliary load balancing loss $\mathcal{L}_{LB}$ was set to 0.01. (5) Unless otherwise specified, the MoE architecture employed a total of 8 experts per module, with 2 experts activated per pass.

We address several key questions regarding our implementation and design choices in the following discussion:

- **What is the value of k in the top-k selection? How is it selected?** Currently, k is treated as a hyperparameter, with two main considerations. First, aligning with other MoE architectures, the number of selected experts typically maintains fixed and sparse, with 2 out of 8 being a common configuration. Second, our framework facilitates expert collaboration, so the top k is not a critical parameter for fully leveraging expert potential.

- **Is the top-k selection in the routing operation (Equation 4) not differentiable?** Regarding the differentiability issue of the top-k in Equation 4, it is important to note that while the top-k operation itself is non-differentiable, it does not directly impact the gradient calculations. Rather,

[3] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

[4] The actual hidden size is rounded to the nearest integer using the Python function "int()".

the fixed number of experts chosen through the top-k operation contribute to the subsequent gradient computations.

- **Are some experts learning to do the initial processing, while others more frequently do the later processing?** In this study, the MoE is applied within each transformer block, so the experts do not possess specific physical significance. Therefore, we have not conducted an analysis of individual expert usage patterns or preferences within the reasoning process.