Exploration-Driven Reinforcement Learning for Expert Routing Improvement in Mixture-of-Experts Language Models

Gyunyeop Kim **Sangwoo Kang**[*]
Department of Computing, Gachon University
`{gyop0817, swkang}@gachon.ac.kr`

Abstract

The performance of MoE-based LLMs depends on the router's ability to select suitable experts; however, the router is typically not explicitly supervised to acquire this routing ability. We propose Exploration-Driven Reinforcement Learning (ERL), which explicitly optimizes the router by exploration of alternative routing paths. For every input, ERL evaluates by (i) the original routing path and (ii) paths in which an α-fraction of routing decisions is randomly perturbed, and treats their performance gap as an advantage signal in a reinforcement learning. Moreover, MoE-ERL$_{wPL}$ mitigates the risk of performance collapse caused by routing reinforcement learning–induced expert overspecialization by intentionally enforcing overlap in experts' knowledge. Without adding parameters or external reward models, our method improves summarization (SAMSum, XSUM), question answering (SQuAD), and language modeling (WikiText-2), and raises routing quality, delivering up to 8.9 × higher MRR than baselines over 100 perturbed routing paths. Code is available at our github[1].

1 Introduction

The rapid advancement of large language models (LLMs) has significantly improved artificial intelligence, with conversational systems such as ChatGPT (OpenAI, 2022) have already brought significant changes to our daily lives (Bommasani and et al, 2021). Empirical studies show scaling training data and parameters to billions—even trillions—consistently leads to improvements in AI performance. (Kaplan et al., 2020; Henighan et al., 2020). However, increasing model size incurs higher memory consumption and computational costs. To alleviate this, recent LLMs like DeepseekMoE (Dai et al., 2024) and Mixtral (Jiang et al., 2024) have adopted the Mixture-of-Experts (MoE) architecture (Shazeer et al., 2017). MoE dynamically activates only a subset of experts per input, enabling conditional sparse computation. This approach maintains large model capacity while significantly improving computational efficiency (Liu et al., 2025; Huang et al., 2024a).

Method	α	**SAMSum**	**XSUM**
MoE	0	26.68	20.05
+random routing	0.25	26.21	18.49
+random routing	0.50	25.52	16.02
+random routing	0.75	23.39	12.07
+random routing	1.00	18.85	06.80

Table 1: Summarization performance(ROUGE-2) on SAMSum and XSUM when an α fraction of the MoE layer's routing decisions are ignored and tokens are assigned to random experts in a Switch Transformer-base-8.

The Switch Transformer (Fedus et al., 2022) extends the standard Transformer (Vaswani et al., 2017) by applying a Mixture-of-Experts architecture. Each feed-forward network (FFN) layer is split into multiple experts, and a learned router selects the most suitable expert for each token. During training, it combines the language modeling loss with a z-loss and an auxiliary load-balancing loss to encourage balanced expert utilization. Despite providing no explicit supervision for expert selection, most previous MoE studies nonetheless rest on the unvalidated assumption that jointly minimizing these three losses is sufficient for the router to implicitly learn optimal routing decisions. However, MoE performance heavily depends on the router's ability to accurately select appropriate experts. Our experiments (Table 1) confirm that randomly perturbing routing decisions markedly degrades performance, highlighting MoE models' vulnerability to suboptimal expert selection. Although previous approaches have attempted to im-

[*]Corresponding author.
[1]https://github.com/KimGyunYeop/MoE-ERL

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 23592–23605
November 4-9, 2025 ©2025 Association for Computational Linguistics

prove routing quality, they mainly rely on heuristics, additional side information, or auxiliary models rather than directly optimizing the routing probability, as detailed in Appendix D.

In this work, we explicitly optimize the router using reinforcement learning to select optimal experts at each token and layer. Specifically, we encourage the MoE-based LLM to explore multiple alternative routing paths and compute an advantage signal based on how each path impacts prediction performance. Using this advantage, we directly train the existing router to enhance its routing accuracy. Crucially, our method introduces no additional parameters or external models, achieving improved MoE-based LLM performance solely through more suitable expert selection. Furthermore, because we leave the standard single-layer MLP router—adopted by most MoE-based LLMs—completely unchanged in structure and mechanism, our approach can be seamlessly integrated into most pretrained MoE models without any architectural modification.

Many prior studies applying reinforcement learning to LLMs (Ryu et al., 2024; Böhm et al., 2019; Ouyang et al., 2022) typically train separate reward models or rely on external evaluation sources such as APIs or human judgments, thereby increasing model size and pipeline complexity. In contrast, our approach computes reward signals directly from the router's own prediction probabilities, allowing the model to autonomously assess performance and explore alternative routing paths without introducing additional modules or external evaluators.

2 Background

2.1 Mixture-of-Experts

Mixture-of-Experts (MoE) (Fedus et al., 2022) arranges multiple expert modules in parallel to enhance computational efficiency and scalability. Each MoE layer is composed of N parallel experts $E_1, E_2, \ldots, E_N$, and, depending on the input, activates only a sparse subset expert. A router G selects the suitable experts for each input token. Formally, the output of an MoE layer is defined as:

$$MoE(h) = \sum_{i=1}^{N} G_i(h) E_i(h) \tag{1}$$

Here, $h \in \mathbb{R}^d$ is the input vector to the MoE layer, and $G_i(h)$ denotes the routing weight assigned by the router to the i-th expert E_i. The router activates only the top-K experts by retaining their weights and sets the others to zero. Formally, $G_i(h)$ is computed as follows:

$$P = Softmax(W_r \cdot h) \tag{2}$$

$$G_i(h) = \begin{cases} \frac{P_i}{\sum_{j \in \text{topK}(P)} P_j} & i \in \text{topK}(P) \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Here, $P \in \mathbb{R}^N$ is the vector of expert activation probabilities, computed by projecting input h through a learnable matrix $W_r \in \mathbb{R}^{N \times d}$.

By leveraging conditional sparse computation, MoE models significantly increase model capacity while maintaining active parameters and computations per token comparable to dense models. Typically, Transformer feed-forward network (FFN) layers are replaced with MoE layers to utilize this benefit.

2.2 Clipped Surrogate Objective of RL

Policy gradient–based (Sutton et al., 1999) reinforcement learning updates the policy π_θ to maximize expected rewards through agent–environment interactions, guided by the advantage signal (Sutton and Barto, 2018). The advantage—defined as the reward benefit of an action over a baseline—quantifies how much better or worse an action performed, so positive advantages are reinforced and negative ones discouraged of action.

In this work, we utilize the Clipped Surrogate Objective from Proximal Policy Optimization (PPO)(Schulman et al., 2017). PPO simplifies computation while retaining the stability of Trust Region Policy Optimization (TRPO)(Schulman et al., 2015). The Clipped Surrogate Objective constrains policy updates to prevent excessive changes, thus ensuring stable and incremental learning. This technique is also employed in many PPO variants, such as Group Relative Policy Optimization (GRPO) (Zhihong Shao, 2024).

The reinforcement learning objective in this work aims to maximize the Clipped Surrogate Objective O^{clip}, defined as follows:

$$O^{\text{clip}}(\theta) = \mathbb{E}_c \left[\min\left(r_c(\theta) A_c,\ \text{clip}(r_c(\theta), 1-\epsilon, 1+\epsilon) A_c \right) \right], \tag{4}$$

$$r_c(\theta) = \frac{\pi_\theta(a_c | s_c)}{\pi_{\theta_{\text{old}}}(a_c | s_c)} \tag{5}$$

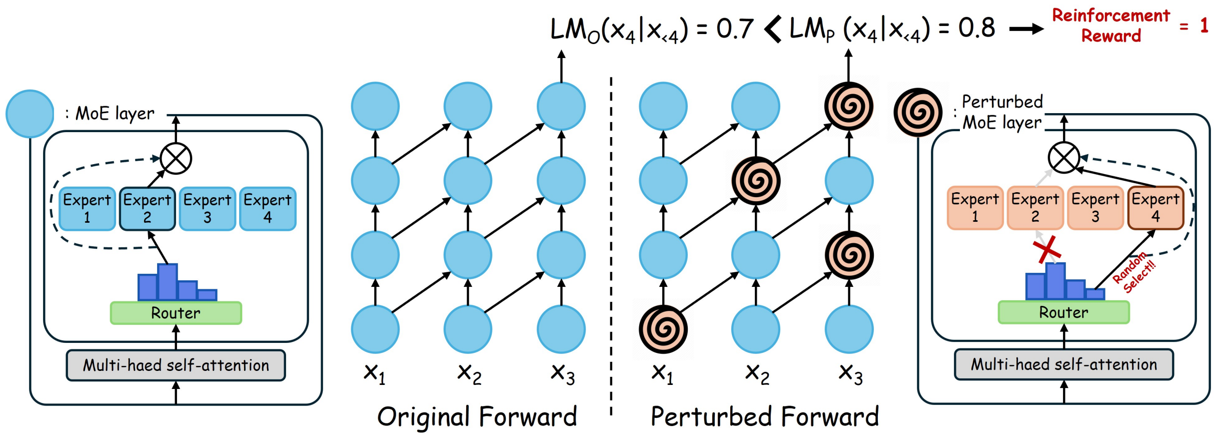

Figure 1: The overall procedure of Exploration-Driven Reinforcement Learning

Here, $\pi_\theta(a_c \mid s_c)$ denotes the action probability under the current policy, and $\pi_{\theta_{\text{old}}}(a_c \mid s_c)$ represents the corresponding probability under the previous policy, used to stabilize updates. The clipping parameter ϵ bounds the change in the probability ratio to prevent abrupt policy shifts. A_c is the advantage estimate, encouraging the policy to increase $\pi_\theta(a_c \mid s_c)$ when $A_c > 0$ and decrease it when $A_c < 0$.

3 Methodology

Many MoE-based LLMs assume that the router will autonomously learn to select suitable experts. However, existing methods do not explicitly train the router for optimality, which makes it challenging to reliably achieve optimal expert selection. So, We propose Exploration-Driven Reinforcement Learning (ERL), which explicitly trains the router toward improved expert selection. ERL explores perturbed (alternative) routing paths and compares their performance against the original path. If an perturbed routing path yields better predictions, the router is reinforced perturbed routing decision; otherwise, the original routing is strengthened. The overall reinforcement learning procedure is illustrated in Figure 1.

3.1 Original vs. Perturbed Forward

Given an input sequence $X = \{x_1, x_2, \dots, x_t\}$, the model explores multiple expert routing paths via two types of forward .

Original forward: At each MoE layer, we select the top-K experts based on the original router's probability distribution. It perform the standard MoE computation described in Section 2.1.

Perturbed forward: To encourage exploration, we deliberately disrupt a subset of routing decisions. Given a sequence of length t passing through l MoE Transformer blocks, a total of $t \times l$ routing decisions occur. We randomly select an α fraction $(0 < \alpha \leq 1)$ of these routing decisions and replace them with perturbed routings. In perturbed routing, the router's original decisions are overridden, and experts are selected randomly. This perturb encourages the model to explore alternative expert routes.

For clarity, we denote the standard MoE layer computation as $MoE(h)$ and the perturbed MoE layer as $MoE^p(h)$. The computation of a single perturbed MoE layer is defined as follows:

$$MoE^p(h) = \sum_{i=1}^{N} G_i^p(h) E_i(h) \quad (6)$$

$$p_i^p \sim \text{Uniform}(0,1), i = 1, 2, .., N \quad (7)$$

$$P^p = (p_1^p, p_2^p, ..., p_N^p) \quad (8)$$

$$G_i^p(h) = \begin{cases} \frac{1}{K} & i \in topK(P^p) \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

To implement random selection, we assign each expert a score sampled from a uniform distribution and activate the top-K experts based on these scores. Here, $P^p \in \mathbb{R}^N$ denotes the vector of random scores assigned to each expert, and $G^p(h) \in \mathbb{R}^N$ represents the weight for each expert, uniformly set based solely on the K, independent of the input h or the router's output.

In this study, for each input sequence X, we perform one original forward and m distinct perturbed forward, each with independently applied perturbations.

3.2 Advantage Strategy

For each perturbed forward, we compute an **advantage**, the primary reinforcement learning signal indicating the relative benefit of an action. In our setting, the **action** corresponds to a perturbed routing decision, and the **policy** is the router G. The advantage is computed independently at each token generation step. If the perturbed forward assigns a higher probability to the correct token prediction than the original forward, the advantage is set to $+1$; if lower, -1; and 0 if equal. Formally, the advantage $\tilde{A}_j$ for the j-th token is defined as:

$$\tilde{a}_j = LM_p(x_j|x_{<j}) - LM_o(x_j|x_{<j}) \tag{10}$$

$$\tilde{A}_j = \begin{cases} 1 & \tilde{a}_j > 0 \\ 0 & \tilde{a}_j = 0 \\ -1 & \tilde{a}_j < 0 \end{cases} \tag{11}$$

In these equation, $LM_o(x_j \mid x_{<j})$ denotes the probability assigned to the correct token x_j by the original forward , while $LM_p(x_j \mid x_{<j})$ is the corresponding probability produced by the perturbed forward.

3.3 Training loss

We define the reinforcement learning loss for the perturbed forward by taking the negative of the objective in Section 3.2, converting it into a minimization problem.

The loss is designed to adjust the routing probabilities G_i(i.e., it same as adjust P_i; see Eq.2 and Eq.3) of randomly selected experts in each perturbed MoE layer based on their advantage values $\tilde{A}$—increasing them when $\tilde{A} > 0$ and decreasing them when $\tilde{A} < 0$. Formally, the reinforcement learning loss function $f(G, \tilde{A})$ for a single perturbed routing decisions is defined as:

$$r_i = \frac{G_i(h)}{G_i^{\text{old}}(h)} \tag{12}$$

$$f(G, \tilde{A}) = -\frac{1}{K} \sum_{i \in \text{top}K(P^p)} \min\!\big(r_i\,\tilde{A}, \text{clip}(r_i, 1-\varepsilon, 1+\varepsilon)\,\tilde{A}\big). \tag{13}$$

Here, $G_i^{\text{old}}(h)$ denotes the expert selection probability under the previous policy. By directly incorporating the advantage signal to action policy, the loss function explicitly optimizes the router toward selecting more suitable experts.

A single perturbed forward involves $t \times l$ routing decisions, of which $\alpha \times t \times l$ are perturbed. An advantage is independently assigned to each perturbed routing decision. Specifically, if a perturbed routing decision occurs at the k-th Transformer block during prediction of the j-th token, only the corresponding $\tilde{A}_j$ is used. Thus, each perturbed routing decision has its own reinforcement learning loss function $f(G^{j,k}, \tilde{A}_j)$. The overall reinforcement learning loss L^{RL} is computed as the average loss across all perturbed routing decisions.

$$L^{RL} = \frac{1}{|S|} \sum_{j,k \in S} f(G^{j,k}, \tilde{A}_j) \tag{14}$$

$$L^p = L^{RL} \tag{15}$$

Here, $S = \{(j_1, k_1), \ldots, (j_M, k_M)\}$ denotes the set of perturbed routing positions, where $j_m \in [1, l]$, $k_m \in [1, t]$, and $|S| = M \approx \alpha lt$. $G^{j,k}$ denotes the router at the k-th Transformer block during generation of the j-th token. Using only the advantage $\tilde{A}_j$ corresponding to the exact perturbation step yields a finer-grained credit assignment and lowers the variance of the RL updates.

The training loss for the original forward , L^o, follows the Switch Transformer formulation(Fedus et al., 2022), comprising the LM loss, z-loss, and auxiliary load-balancing loss. We combine this with the perturbed forward loss L^o, scaled by a weighting factor δ. The overall training loss is:

$$L = L^o + \delta L^p \tag{16}$$

3.4 LM Loss of Perturbed Forward

As reinforcement learning progresses, the router converges to selecting the "most suitable" expert for each input. Although this promotes expert specialization on specific information types, during training the router may consistently select the same expert for all inputs requiring a given information type I. While such specialization allows each expert to specialize in its domain, it also significantly reduces knowledge overlap among experts. Consequently, if the router mistakenly selects an inappropriate expert, there is an increased risk that the expert will fail to adequately process the input. We refer to this vulnerability as the "expert misclassification risk" of routing.

To address the expert misclassification risk, we incorporate an auxiliary loss—the language modeling loss of perturbed forward (L^{pLM}). This encourages randomly activated experts to learn similar information, thereby preserving a degree of knowledge overlap. As a result, even when a suboptimal expert is selected, performance degradation can be alleviated. The combined perturbed forward loss in this setting is defined as:

$$L^{p}_{wPL} = L^{RL} + L^{pLM} \tag{17}$$

This auxiliary loss explicitly promotes knowledge overlap by encouraging different experts to learn from the same data. While various mechanisms could be used to induce such overlap, we adopt this simple strategy to clearly assess whether reinforcement learning enables the router to discover optimal routing path.

4 Experiment

4.1 Task and Datasets

Summarization. SAMSum (Gliwa et al., 2019) consists of 15k dialogue–summary pairs from everyday conversations. **XSUM** (Narayan et al., 2018) contains approximately 227k BBC news articles paired with single-sentence summaries.

Question Answering. SQuAD (Rajpurkar et al., 2016) is an extractive QA dataset comprising approximately 98k question–answer pairs derived from Wikipedia passages.

Language Modeling. We evaluate language modeling performance using **WikiText-2** (Merity et al., 2016), approximately 2.5M token corpus commonly used for small- to medium-scale models.

These three tasks enable us to assess the generalizability and effectiveness of our reinforcement learning method across diverse settings. Dataset statistics are provided in Appendix A.

4.2 Baselines

In this work, we compare with following three baseline configurations: **(1) Dense.** A fully dense model with MoE disabled; only a single expert is used, and no sparse computation is performed. **(2) MoE.** A standard MoE model with routing. We adopt the Switch-base-8 configuration. **(3) MoE-share.** A shared-expert variant based on Deepseek-MoE (Dai et al., 2024), where one shared expert is activated for all inputs with router-selected expert.

Summarization and QA Baselines. For summarization and QA tasks, all baselines are fine-tuned using pretrained Switch-base-8(Fedus et al., 2022).

MoE. The original Switch Transformer without modification. **Dense.** Sparse computation is disabled by retaining only a single expert and removing the router. **MoE-share.** We replicate one expert from the MoE configuration to create a shared expert that is activated for every input.

Language Modeling Baselines. For language modeling tasks, which use decoder-only architectures, we construct three baselines based on pretrained GPT-2 (Radford et al., 2019), following the Hyper-MoE setup (Zhao et al., 2024):

Dense. The original GPT-2 without modification. **MoE.** The GPT-2 FFN layer is replicated to form eight parallel experts, with a initialized router enabling sparse computation. **MoE-share.** Augment the MoE configuration with a shared expert created by replicating the FFN layer.

To ensure a fair comparison, we use the top-1 routing configuration from switch-base-8 and match all other MoE hyperparameters (number of experts, expert capacity, etch) to it. Since deep learning performance also depends on hyperparameters, model size and experiment enviorments, we re-evaluate both the baseline and our proposed method in our experiments.

4.3 Experiment Setting

For summarization and QA tasks, we used identical hyperparameters across all experiments. For language modeling, a few hyperparameters were adjusted. All hyperparameters were selected via greedy search on SAMSum and WikiText-2, and applied consistently across datasets. Reported results reflect the best performance among epochs 5, 10, and 15.

To address training instability when adapting GPT-2 to an MoE setting, we initialized router parameters from $\mathcal{N}(0, 0.1)$, following Fedus et al. (2022). Additional implementation details are provided in Appendix B.

We apply our proposed method to MoE baseline configuration and reinforcement the model accordingly. Since advantage computation depends on next-token prediction probabilities, all experiments and analyses in this paper focus exclusively on the

Model	#Exp.	#Act. Exp.	SAMSum			XSUM		
			R-1 (↑)	**R-2** (↑)	**R-L** (↑)	**R-1** (↑)	**R-2** (↑)	**R-L** (↑)
Dense	–	–	51.85	26.61	42.61	42.98	19.89	34.96
MoE	8	1	51.70	26.68	42.89	42.97	20.05	35.00
MoE-share	9	2	51.64	26.65	42.92	42.90	19.93	34.87
HyperMoE (2024)	8	1	51.50	26.84	43.01	-	19.67*	-
MoE-ERL	8	1	<u>51.98</u>	<u>26.94</u>	<u>43.11</u>	<u>43.10</u>	<u>20.13</u>	<u>35.05</u>
MoE-ERL$_{wPL}$	8	1	**52.29**	**27.20**	**43.47**	**43.26**	**20.25**	**35.27**

Table 2: Downstream task performance on the SAMSum and XSUM (**ROUGE** scores (%); higher is better). Within each column, the **bold** numbers denote the best result and the <u>underlined</u> numbers denote the second-best. * denotes scores reported in the original paper.

Model	SQuAD		WIKI2
	EM (↑)	**F1** (↑)	**ppl.** (↓)
Dense	83.11	90.44	<u>20.53</u>
MoE	82.80	90.37	21.20
MoE-share	83.03	90.31	20.70
HyperMoE(2024)	**84.6***	-	21.49*
SimSMoE(2025)	82.80*	-	-
MoE-ERL	83.16	<u>90.50</u>	<u>20.53</u>
MoE-ERL$_{wPL}$	<u>83.26</u>	**90.63**	**20.44**

Table 3: Downstream task performance on SQuAD (Exact Match & F1 score; higher is better) and WikiText-2 perplexity (lower is better).

decoder. We refer to our proposed method as **MoE-ERL**, and its variant incorporating the perturbed forward LM loss (Section 3.4) as **MoE-ERL$_{wPL}$**.

5 Main Results

Tables 2–3 indicate that MoE-ERL and MoE-ERL$_{wPL}$ outperform conventional baselines on the majority of datasets.

Summarization. On both SAMSum and XSUM, **MoE-ERL** outperforms all baselines across ROUGE metrics. In ROUGE-2, it improves over the strongest baseline by 0.26 (SAMSum) and 0.08 (XSUM). The variant **MoE-ERL$_{wPL}$** achieves improves of 0.52 and 0.20, respectively. These results demonstrate that reinforcement learning improves the summarization capability of MoE models, and incorporating the LM loss for perturbed forwards yields additional benefit.

Question Answering. On SQuAD, MoE-ERL$_{wPL}$ achieves modest gains of 0.14 EM and 0.06 F1 over the best baseline. While improvements are smaller than in other tasks and do not surpass HyperMoE, this is likely due to the limited effect of decoder-side routing in extractive QA, where answers are typically short.

Language Modeling. On WikiText-2, the Dense baseline achieved relatively strong performance, likely due to its direct use of GPT-2's pretrained dense weights without modify architecture. Despite this, **MoE-ERL** reduced perplexity by 0.67 compared to the MoE baseline, achieving performance on same with Dense. **MoE-ERL$_{wPL}$** further improved perplexity by 0.09, resulting in a total reduction of 0.76 over MoE.

MoE-ERL consistently matches or outperforms the performance of Dense, MoE, and MoE-share baselines. In all cases, the LM-loss variant (**MoE-ERL$_{wPL}$**) further improves upon MoE-ERL. Supplementary experiments confirm that these improvement hold across varying number of expert (Appendix C).

6 Analysis

6.1 Analysis of Routing

To evaluate whether the RL-trained router selects improved routing paths, we conduct an additional analysis. For each fine-tuned model, we record the ROUGE-2 score (summarization) or perplexity (language modeling) from the original forward. We then generate 99 perturbed forward per data by randomly altering decoder routing 33 times at each $\alpha \in 0.25, 0.50, 0.75$, yielding 100 scores per data. We report the rank of the original path using Mean Reciprocal Rank (MRR) and Hit@1 (the proportion of cases where the original forward ranks first). Evaluations are conducted on 300 randomly

Model	SAMSum		XSUM		WIKI2	
	MRR (↑)	Hit@1 (↑)	MRR (↑)	Hit@1 (↑)	MRR (↑)	Hit@1 (↑)
MoE	0.11	6.8	0.06	2.0	0.05	2.9
MoE-ERL	**0.35**	**18.4**	**0.37**	**17.8**	**0.17**	**9.5**
MoE-ERL$_{wPL}$	0.16	11.7	0.10	5.0	0.14	8.0

Table 4: Routing quality - original routing path vs. 99 perturbed routing path. Reported are Mean Reciprocal Rank and Hit@1 (higher is better) of original routing path.

sampled data per dataset, with results shown in Table 4.

MoE vs. MoE-ERL. For baseline MoE, the original routing path rarely ranked highly, with MRRs of 0.11, 0.06, and 0.05, and Hit@1 scores of 6.8%, 2.0%, and 2.9% across the datasets. This indicates that conventional routers often fail to select optimal routing paths. In contrast, **MoE-ERL** achieved significantly higher MRRs of 0.35, 0.37, and 0.17, and Hit@1 scores of 18.4%, 17.8%, and 9.5%, improving the likelihood of optimal selection by 2.7–8.9×. These results confirm that reinforcement learning substantially enhances routing quality.

MoE-ERL$_{wPL}$ shows lower MRR and Hit@1 scores compared to MoE-ERL. We estimate this is because intentionally increasing knowledge overlap among experts, through the additional LM loss of perturbed forwards, raises the likelihood that even suboptimal routing paths achieve acceptable performance. Nevertheless, MoE-ERL$_{wPL}$ still outperformed the MoE baseline by improving both metrics approximately 1.5–2.75×, clearly demonstrating that it effectively enhances routing quality.

6.2 Expert Misclassification Risk of ERL

We analyze the "expert misclassification risk" described in Section 3.4. Specifically, we compare our proposed methods to the MoE baseline under experimental settings similar to those in Table 5. We measure the performance degradation of fine-tuned models when perturbing decoder routing decisions at inference time, with varying fractions α. Unlike the experiment presented in Table 1, here we perturb only in the decoder.

Baseline (MoE). In our experiments, the standard MoE baseline exhibited minimal degradation under decoder-only perturbations. Some performance drops became when the perturbation ratio α exceeded 0.75. On XSUM, scores declined by 3.8 and 6.15 for $\alpha = 0.75$ and $\alpha = 1.00$.

Model	α	SAMSum	XSUM	SQuAD
MoE	0.00	26.68	20.05	82.80
	0.25	27.10	19.31	82.81
	0.50	26.42	18.14	82.61
	0.75	26.05	16.25	82.59
	1.00	25.11	13.90	82.13
MoE-ERL	0.00	26.94	20.13	83.16
	0.25	14.19	4.73	31.95
	0.50	2.93	0.24	10.49
	0.75	0.56	0.01	2.90
	1.00	0.02	0.00	0.36
MoE-ERL$_{wPL}$	0.00	27.20	20.25	83.26
	0.25	26.99	20.14	83.11
	0.50	27.01	20.10	83.03
	0.75	27.18	20.02	82.95
	1.00	27.01	19.84	82.99

Table 5: Impact of randomly confusing a fraction α of decoder's routing. Scores are reported for SAMSum/XSUM (ROUGE-2) and SQuAD (EM)

MoE-ERL. In contrast, **MoE-ERL** exhibited substantial performance degradation starting from $\alpha = 0.25$, consistently across all datasets. As anticipated, this suggests that reinforcement learning reduces knowledge overlap among experts, increasing the expert misclassification risk and leading to performance drops under suboptimal routing.

MoE-ERL$_{wPL}$. In **MoE-ERL**wPL, explicitly augmenting expert overlap mitigated performance degradation under routing perturbations, maintaining stable results with only minor drops (0.1–0.4 points). This robustness likely stems from the perturbed LM loss, which encourages knowledge sharing among experts. AOverall, the results here and in Table 4 show that **MoE-ERL**wPL both enhances the router's ability to select optimal routing paths and reduces the risk of expert misclassification.

6.3 Analysis of Routing Confidence

To assess the model's confidence in its routing decisions, we measured the probability of selecting the top-1 expert, $G_{\text{top1}}(h)$, and visualized the distribution using histograms (Figure 2). For **MoE-**

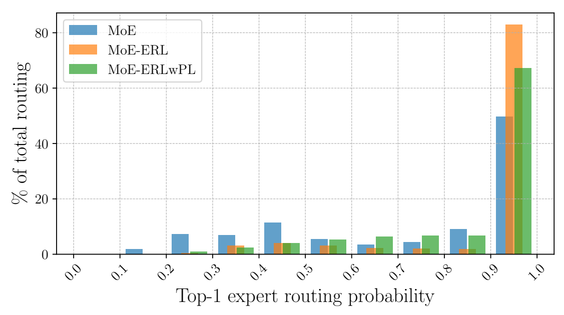

Figure 2: Histogram of top-1 expert routing probability on SAMSum.

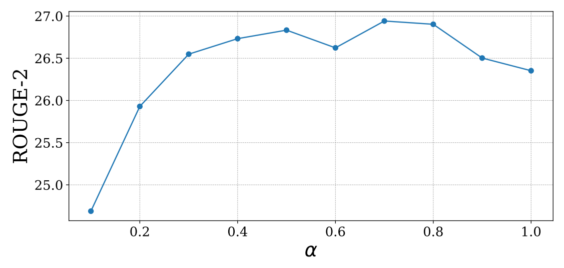

Figure 3: Effect of the perturbed routing decision ratio (α) during training on SAMSum.

ERL, approximately 80% of routing decisions had a top-1 probability above 0.9, indicating high confidence—about 30 percentage points higher than the baseline MoE. This suggests that reinforcement learning significantly enhances routing certainty. In comparison, **MoE-ERL**$_{wPL}$ showed around 67% in the same threshold, higher than MoE but lower than MoE-ERL, reflecting its induced expert overlap.

6.4 Ablation Study

Figure 3 presents SAMSum ROUGE-2 scores for **MoE-ERL** models trained with different perturbation rates $\alpha \in [0.1, 1.0]$. Performance remains stable for α values between 0.3 and 0.8, but drops sharply when α is too small or too large. We estimate Low α results in insufficient exploration, limiting the benefit of reinforcement learning, while high α introduces excessive randomness, weakening the learning signal and degrading performance.

We additionally conduct an ablation study on the advantage strategy for reinforcement learning. Our proposed approach, referred to as the *static advantage strategy* (Equations 10–11), assigns a static value of $+1$, 0, or -1 depending on whether the perturbed outperforms the original forward. As an alternative, we implement a *log advantage strategy*, which uses the log-difference between the perturbed and original token prediction probabilities:

Model	**Advantage Strategy**	**SAMSum**	**Wiki2**
MoE-ERL	static	**26.94**	**20.53**
MoE-ERL	log	26.64	20.55
MoE-ERL$_{wPL}$	static	**27.20**	**20.44**
MoE-ERL$_{wPL}$	log	26.90	20.45

Table 6: Ablation—static vs. log advantage. Reported are SAMSum ROUGE-2 (higher is better) and WikiText-2 perplexity (lower is better)

$$\tilde{A}_j = \log(P_p(x_j|x_{<j})) - \log(P_o(x_j|x_{<j})) \quad (18)$$

While the log advantage strategy captures fine-grained differences between original and perturbed outputs, the static strategy provides clearer sample-level ranking, even under small performance gaps.

The results in Table 6 show that the static advantage strategy consistently outperforms the log-based variant across all experiments. We observed that the log advantage strategy frequently caused loss divergence during training. We attribute this issue to the nature of the logarithmic operation, which can yield extremely large advantage values, thereby destabilizing training and resulting in divergence.

7 Conclusion

We proposed **MoE-ERL**, a reinforcement learning method that explicitly optimizes expert routing in MoE-based LLMs without introducing additional parameters or external reward models. MoE-ERL leverages the performance gap between original and perturbed routing paths as an advantage signal to guide the router toward optimal expert selection. We further proposed **MoE-ERL**$_{wPL}$, which incorporates an auxiliary LM loss on perturbed routing paths to mitigate the risk of over-specialization.

Most of experiments on SAMSum, XSum, SQuAD, and WikiText-2 demonstrated that both variants outperform Dense, MoE, and MoE-share baselines, improving routing quality (MRR and Hit@1) by up to 8.9×. We also identified the expert misclassification risk due to reduced knowledge overlap, which **MoE-ERL**$_{wPL}$ effectively alleviates. Future work may explore additional strategies to further reduce misclassification risk while maintaining routing precision.

Limitations

The reinforcement learning proposed in this study is only applicable to decoder models. MoE-ERL calculates advantages by comparing token-level prediction probabilities between original and perturbed forward , naturally aligning the t token-level advantages with corresponding MoE routing decisions. However, encoder models typically provide a single document- or sentence-level classification probability, resulting in only one overall advantage value ($\tilde{A}$). allocating this single advantage across all perturbed MoE layer and token is highly challenging. If one resorts to naively broadcasting the same value uniformly across all perturbed MoE layers and tokens, the credit signal becomes so diluted that it can no longer reveal which specific tokens or MoE layers are responsible for the observed performance gains, thereby impeding effective learning. Thus, future research should explore designing effective token-level reward schemes and credit assignment strategies specifically tailored to encoder-based MoE architectures.

Additionally, our proposed method incurs higher training costs. Typically, standard deep learning methods perform one forward and one backward per training step. In contrast, our method requires additional forward passes to compute results from the perturbed forwards. Specifically, for each backpropagation step, the method performs $1 + m$ forward passes (depending on the number of perturbed forwards m) while keeping a single backward pass, thereby increasing the training cost.

To complement wall-clock time, we also report a hardware-independent efficiency metric in terms of Training FLOPs. Let F_{fwd} and F_{bwd} denote the forward and backward FLOPs of a baseline step. Standard training requires $F_{\text{fwd}}+F_{\text{bwd}}$, whereas our method requires $(1 + m \cdot \phi)\, F_{\text{fwd}} + F_{\text{bwd}}$. Here, ϕ represents the fraction of forward FLOPs attributed to the decoder ($\phi = 1$ for decoder-only models, where cost scales with the decoder input length T_{dec}; for encoder–decoder models, $\phi \approx \frac{T_{\text{dec}}}{T_{\text{enc}}+T_{\text{dec}}}$, where T_{enc} and T_{dec} denote the number of encoder and decoder input tokens, respectively).

Under our experimental setting ($m = 3$), the measured wall-clock training times confirmed this trend: encoder–decoder tasks (SQuAD, SAMSum, and XSUM), where only decoder forwards are multiplied, required approximately $1.2\times$, $1.6\times$, and $1.8\times$ longer, proportional to the generated sequence length T_{dec}. For the decoder-only language model (WikiText-2), where the entire model requires multiple forwards, training required approximately $2.8\times$ longer. The additional inclusion of wPL introduced minimal differences in training time. Importantly, this extra overhead occurs only during training, while inference remains unaffected, requiring just the original forward identical to conventional MoE models. In summary, theoretical FLOPs analysis and measured wall-clock times are consistent, highlighting that the main efficiency trade-off of our method lies in the increased number of forward passes.

Acknowledgements

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2022R1A2C1005316)

References

Florian Böhm, Yang Gao, Christian M. Meyer, Ori Shapira, Ido Dagan, and Iryna Gurevych. 2019. Better rewards yield better summaries: Learning to summarise without references. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3110–3120, Hong Kong, China. Association for Computational Linguistics.

Rishi Bommasani and Drew A. Hudson et al. 2021. On the opportunities and risks of foundation models. *ArXiv*.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.x. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.k. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, Bangkok, Thailand. Association for Computational Linguistics.

Nishanth Dikkala, Nikhil Ghosh, Raghu Meka, Rina Panigrahy, Nikhil Vyas, and Xin Wang. 2023. On the benefits of learning to route in mixture-of-experts models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9376–9396, Singapore. Association for Computational Linguistics.

Giang Do, Hung Le, and Truyen Tran. 2025. SimSMoE: Toward efficient training mixture of experts via solving representational collapse. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2012–2025, Albuquerque, New Mexico. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Preprint*, arXiv:2101.03961.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China. Association for Computational Linguistics.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M. Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. 2020. Scaling laws for autoregressive generative modeling. *Preprint*, arXiv:2010.14701.

Haiyang Huang, Newsha Ardalani, Anna Sun, Liu Ke, Shruti Bhosale, Hsien-Hsin S. Lee, Carole-Jean Wu, and Benjamin Lee. 2024a. Toward efficient inference for mixture of experts. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. 2024b. Harder task needs more experts: Dynamic routing in MoE models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12883–12895, Bangkok, Thailand. Association for Computational Linguistics.

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, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Jiacheng Liu, Peng Tang, Wenfeng Wang, Yuhang Ren, Xiaofeng Hou, Pheng-Ann Heng, Minyi Guo, and Chao Li. 2025. A survey on inference optimization techniques for mixture of experts models. *Preprint*, arXiv:2412.14219.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Tam Nguyen, Ngoc N. Tran, Khai Nguyen, and Richard G. Baraniuk. 2025. Improving routing in sparse mixture of experts with graph of tokens. *Preprint*, arXiv:2505.00792.

OpenAI. 2022. Introducing ChatGPT. `https://openai.com/blog/chatgpt`. Accessed: 2025-05-17.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Zihan Qiu, Zeyu Huang, Shuang Cheng, Yizhi Zhou, Zili Wang, Ivan Titov, and Jie Fu. 2025. Layerwise recurrent router for mixture-of-experts. In *The Thirteenth International Conference on Learning Representations*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Jie Ren, Yewen Li, Zihan Ding, Wei Pan, and Hao Dong. 2021. Probabilistic mixture-of-experts for efficient deep reinforcement learning.

Stephen Roller, Sainbayar Sukhbaatar, arthur szlam, and Jason Weston. 2021. Hash layers for large sparse models. In *Advances in Neural Information Processing Systems*, volume 34, pages 17555–17566. Curran Associates, Inc.

Sangwon Ryu, Heejin Do, Yunsu Kim, Gary Lee, and Jungseul Ok. 2024. Multi-dimensional optimization for text summarization via reinforcement learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5858–5871, Bangkok, Thailand. Association for Computational Linguistics.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France. PMLR.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*, second edition. The MIT Press.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yuanhang Yang, Shiyi Qi, Wenchao Gu, Chaozheng Wang, Cuiyun Gao, and Zenglin Xu. 2024. XMoE: Sparse models with fine-grained and adaptive expert selection. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11664–11674, Bangkok, Thailand. Association for Computational Linguistics.

Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. 2024. AdaMoE: Token-adaptive routing with null experts for mixture-of-experts language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6223–6235, Miami, Florida, USA. Association for Computational Linguistics.

Hao Zhao, Zihan Qiu, Huijia Wu, Zili Wang, Zhaofeng He, and Jie Fu. 2024. HyperMoE: Towards better mixture of experts via transferring among experts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10605–10618, Bangkok, Thailand. Association for Computational Linguistics.

Qihao Zhu Runxin Xu Junxiao Song Mingchuan Zhang Y.K. Li Y. Wu Daya Guo Zhihong Shao, Peiyi Wang. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Y Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*.

A Dataset Statistics

In this study, we evaluated our proposed approach across various tasks and datasets. Since our method is applied exclusively to the decoder for text generation, we present detailed statistical information, including the number of tokens used for generation, in Table 9. The datasets, ordered by size, are XSUM, SQuAD, SAMSum, and WikiText-2. We specifically report the number of tokens input to the decoder—the context in which our proposed method operates. Among these datasets, WikiText-2, where entire sentences serve as decoder input, has the longest token length, followed by XSUM, SAMSum, and SQuAD, respectively.

hyperparameter	value
α (sum.&QA)	0.7
α (lm)	0.6
δ (sum.&QA)	2.0
δ (lm)	0.5
m	3
ϵ	0.2
learning rate	1e-5
weight decay	0.1
batch size (sum.&QA)	8
batch size (lm)	4
optimizer	Adam
Beta1,2	0.9,0.999
lr scheduling	linear
n beam	6

Table 7: Detailed Hyperparameter Value. (sum. & QA) denote value for summarization and QA task and (lm) denote value for language modeling task

B Additional Experiment Setting

All hyperparameters used in our experiments are listed in Table 7. Here, α denotes the fraction of routing decisions perturbed in the perturbed forward ; δ controls the weight of the perturbed-forward loss L^p in the overall loss L; m is the number of perturbed forwards generated per original forward; and ϵ is the clipping coefficient in the Clipped Surrogate Objective. Hyperparameters for the WikiText experiments were tuned via greedy search on WikiText-2, while those for all other datasets were tuned on SAMSum and then applied uniformly. We use the HuggingFace `evaluate` library(Wolf et al., 2020) for all evaluation metrics. All main experiments reported in the paper were run on a machine with RTX 4090 $\times$ 2, whereas the supplementary experiments in this appendix were conducted on a machine with RTX 3090 $\times$ 4. Training times for the proposed method were approximately 4h on SAMSum, 60h on XSUM, 20h on SQuAD, and 1.5h on WikiText-2.

To verify the fairness of our comparisons, we report both the total parameter count and the number of parameters actually activated during inference for every model (Table 10). Because MoE-share introduces one additional shared expert per layer, it contains the largest number of parameters; the proposed MoE-ERL and MoE-ERL$_{\text{wPL}}$ have exactly the same capacity as the vanilla MoE, while the Dense baseline is the smallest. The same ordering holds for activated parameters: MoE-share routes two experts per token, whereas MoE, MoE-ERL, and MoE-ERL$_{\text{wPL}}$ route a single expert, and the Dense model, which lacks a router, activates none. These statistics confirm that our performance gains come solely from improved routing quality, not from an increase in model size.

#.Experts	Model	SAMSum (↑)
16	MoE	26.47
	MoE-ERL	26.53
	MoE-ERL$_{wPL}$	**27.20**
32	MoE	26.76
	MoE-ERL	27.17
	MoE-ERL$_{wPL}$	**27.90**

Table 8: Experiment of difference model capacity on SAMSum(ROUGE-2)

C Additional Results

We conducted additional experiments to verify the effectiveness of our proposed method on models with a larger number of experts. Experimental results confirmed that even when increasing the number of experts, our proposed MoE-ERL and MoE-ERL$_{wPL}$ consistently improved ROUGE-2 scores compared to the baseline. This demonstrates the efficacy of our approach across various MoE configurations with differing numbers of experts.

D Related Works

The router is a critical component in Mixture-of-Experts (MoE) architectures, responsible for selecting the most suitable experts. Several studies have explored ways to enhance routing decisions;

Dataset	#data			avg. #token of {summary, answer, document}		
	#train	#dev	#test	#train	#dev	#test
SAMSum	14,732	818	819	25.5	25.5	25.2
XSUM	204,045	11,332	11,334	26.1	26.1	26.1
SQuAD	87,599	10,570	-	4.7	4.4	-
WikiText-2	36,718	3,760	4,358	65.1	65.8	65.0

Table 9: Statistics of the datasets used in our experiments.

pre-trained model	Model	#Exp.	#Act. Exp.	Total Parameter	Act. Parameter
switch-base-8	Dense	-	-	222.94 M	222.94 M
	MoE	8	1	619.34M	222.98M
	MoE-share	9	2	647.65M	251.29M
	MoE-ERL	8	1	619.34M	222.98M
	MoE-ERL$_{wPL}$	8	1	619.34M	222.98M
gpt2	Dense	-	-	124.44 M	124.44 M
	MoE	8	1	322.86 M	124.51 M
	MoE-share	9	2	351.19 M	152.85 M
	MoE-ERL	8	1	322.86 M	124.51 M
	MoE-ERL$_{wPL}$	8	1	322.86 M	124.51 M

Table 10: Model capacities used in our experiments. "# Exp." is the number of experts per MoE layer, "# Act." is activated.

however, as discussed in Dikkala et al. (2023), these efforts have primarily focused on improving computational efficiency. For example, Zhou et al. (2022) addressed load imbalance among experts to enhance efficiency in multi-node settings, while recent approaches like Huang et al. (2024b) and Zeng et al. (2024) dynamically adjusted the number of activated experts, enabling computations strictly as needed.

Beyond efficiency, several approaches have been proposed with the explicit goal of improving routing accuracy. Techniques such as hash-based routing (Roller et al., 2021) and cosine similarity-based allocation in XMoE (Yang et al., 2024) have refined token-expert mappings. However, these methods predominantly rely on non-trainable routing mechanisms that process tokens independently, thus failing to effectively capture interactions between tokens. In contrast, Dikkala et al. (2023) empirically demonstrated that trainable router significantly outperform fixed (non-trainable) routers, with performance gaps widening as the number of experts increases. More recently, Nguyen et al. (2025) utilized token similarity and attention matrices to cluster similar tokens toward the same experts, while RMoE (Qiu et al., 2025) employed shared GRUs across layers to propagate prior routing decisions, thereby improving routing accuracy.

Instead of complicating the router with auxiliary structures, we retain the standard single-layer MLP router—used by virtually most MoE-based LLMs—unchanged and optimize it with reinforcement learning (RL). Concretely, we treat the router's expert-selection probabilities as the policy, define a token-level reward as the confidence gap between the original and a perturbed routing path, and update the router with RL. To the best of our knowledge, no prior work applies an using reinforcement learning of router optimization. Because the algorithm introduces no extra parameters, modules, or inference-time operations, it can be incorporated into most pretrained MoE model with minimal engineering effort, yielding substantial accuracy gains while preserving inference efficiency.

E Probabilistic View

A conditional MoE is a latent-variable model with gating $p_{\theta_g}(z \mid x)$ and experts $p_{\theta_e}(y \mid x, z)$ (Ren et al., 2021):

$$\log p_\Theta(y \mid x) = \log\textstyle\sum_z p_{\theta_g}(z \mid x)\, p_{\theta_e}(y \mid x, z), \quad z \in [K]^{L\times T}$$

For clarity we present the top-1 case (categorical z per position); the top-k case follows analogously.

Using the ELBO with $q_\phi = p_{\theta_g}$ gives the score-function gradient (REINFORCE) for the router:

$$\nabla_{\theta_g}\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{z\sim p_{\theta_g}}\left[\log p_{\theta_e}(y \mid x, z)\, \nabla_{\theta_g} \log p_{\theta_g}(z \mid x)\right]$$

Estimator. We subtract a control variate (baseline) from the router's deterministic top-1 path z^{O}:

$$\hat{g} = \Big(\log p_{\theta_e}(y \mid x, z^{\mathrm{P}}) - \mathrm{sg}\big[\log p_{\theta_e}(y \mid x, z^{\mathrm{O}})\big]\Big)\, \nabla_{\theta_g} \log p_{\theta_g}(z^{\mathrm{P}} \mid x)$$

with $z^{\mathrm{P}} \sim p_{\theta_g}(\cdot \mid x)$. Because the baseline is independent of the sampled z^{P} and treated as a constant w.r.t. θ_g (stop-gradient), the estimator is *unbiased in the on-policy case*.

Stability. We optimize a PPO-style clipped surrogate with ratio $r = \frac{p_{\theta_g}(z^{\mathrm{P}}|x)}{p_{\bar{\theta}_g}(z^{\mathrm{P}}|x)}$ and advantage $A = \log p_{\theta_e}(y \mid x, z^{\mathrm{P}}) - \log p_{\theta_e}(y \mid x, z^{\mathrm{O}})$:

$$\hat{\mathcal{J}}_{\text{PPO}} = \mathbb{E}\big[\min(rA,\ \mathrm{clip}(r, 1-\varepsilon, 1+\varepsilon)A)\big]$$

This surrogate improves optimization stability but does not guarantee exact ELBO ascent.

α-perturbation. For exploration, we also sample routes from

$$q_\alpha = (1-\alpha)\, p_{\bar{\theta}_g} + \alpha\, \rho$$

where the local randomizer ρ has support covering that of p_{θ_g}. An unbiased off-policy update uses importance weights $w = p_{\theta_g}/q_\alpha$; in practice we often rely on PPO clipping with moderate α (e.g., 0.3–0.8) for a favorable bias–variance trade-off.

Takeaway. Our "perturb-and-baseline" update is a control-variate implementation of the ELBO score-function gradient for MoE, stabilized by a trust-region surrogate and light exploration.

Use of AI Assistant

We used ChatGPT(OpenAI; accessed May 19, 2025) solely to translate the draft into English, and all translated content was thoroughly reviewed by the authors.