

# 000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 BALANCING THE EXPERTS: UNLOCKING LoRA-MoE FOR GRPO VIA MECHANISM-AWARE REWARDS

Anonymous authors

Paper under double-blind review

## ABSTRACT

Parameter-efficient Mixture-of-Experts (MoE) architectures, such as LoRA-MoE, enable strong and generalizable fine-tuning. However, a critical problem arises when fine-tuning these architectures with advanced reinforcement learning algorithms such as Group Relative Policy Optimization (GRPO). Traditional supervised techniques are not naturally compatible with the GRPO objective, and naive combinations fail to effectively address routing collapse and the underutilization of MoE adapter parameters. To resolve this disconnect, we introduce Routing-Optimized Group Relative Policy Optimization (RO-GRPO), a mechanism-aware framework. It turns internal expert routing statistics collected during training into a direct reward signal, seamlessly integrating routing supervision into the reinforcement fine-tuning (RFT) process. This enables effective optimization of parameter utilization and improves performance on both unimodal and multimodal mathematical reasoning tasks, all without extra training stages. Our work provides the first demonstration that a scalar reward in GRPO can be engineered from a model’s own internal mechanics to explicitly guide its optimization, extending alignment from mere behavior tuning to holistic mechanism alignment.

## 1 INTRODUCTION

Large language models (LLMs) have significantly advanced many artificial intelligence applications, but their large size poses challenges for practical deployment, especially regarding fine-tuning efficiency (Han et al., 2024; Hu et al., 2021). Among parameter-efficient fine-tuning (PEFT) approaches, LoRA-MoE (Dou et al., 2024), which combines Low-Rank Adaptation (LoRA) (Hu et al., 2021) with a mixture-of-experts (MoE) architecture, has emerged as a particularly promising technique (Li et al., 2024a; Gou et al., 2024; Mu & Lin, 2025).

However, despite its success in supervised fine-tuning (SFT) (Dou et al., 2024; Li et al., 2024a), applying LoRA-MoE to reinforcement learning fine-tuning (RFT) frameworks such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024) presents a key challenge. In SFT, routing is typically guided by an auxiliary load-balancing loss (Fedus et al., 2022; Lewis et al., 2021; Zoph et al., 2022; Dai et al., 2022; Wang et al., 2024a). In GRPO, our goal is to employ a loss-free mechanism that jointly optimizes task performance and routing load through the rollout-time reward signal, rather than uniformly enforcing the same constraint on all routing decisions via an auxiliary routing loss. When GRPO relies solely on the external task reward, the training signal remains blind to internal routing decisions (Omi et al., 2025; Harvey et al., 2025). Without explicit guidance, the routing mechanism often collapses, leading to severe expert imbalance and underutilization of the model’s parametric capacity, which limits the effectiveness of the modular architecture.

To bridge this gap, we propose **RO-GRPO** (Routing-Optimized GRPO), a novel framework that integrates routing-awareness into the RFT process through a carefully designed mechanism reward. Our key insight is that routing statistics collected during generation, such as routing entropy and load distribution, can be transformed into a reward signal that aligns the internal routing mechanism with task performance (Chen et al., 2022; Cong et al., 2024b). Specifically, we augment the standard task reward with two complementary components: one promoting confident routing decisions (low entropy) and another encouraging balanced expert utilization. These routing rewards are integrated into the GRPO objective, enabling a unified optimization process that requires no auxiliary losses or architectural modifications (see Figure 1 for a schematic comparison).

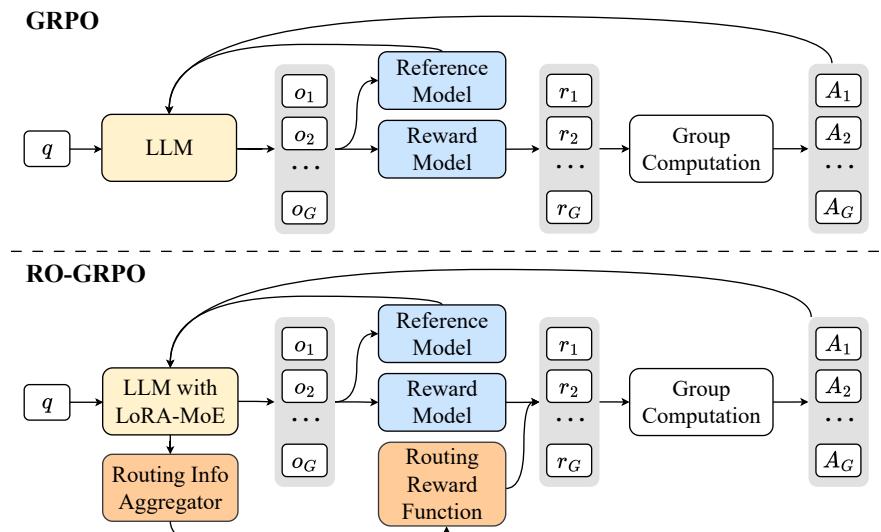


Figure 1: Comparison of standard GRPO and our RO-GRPO.

The main contributions of this paper are as follows:

- To our knowledge, this is the first systematic study of LoRA-MoE architectures within the RFT framework, identifying and addressing key challenges such as routing collapse and expert underutilization that arise during GRPO-based training.
- We propose RO-GRPO, a novel framework that integrates routing statistics directly into the GRPO reward function. This enables the unified optimization of both task performance and internal routing efficiency without requiring auxiliary losses.
- Our method achieves consistent improvements over baselines (e.g., standard LoRA and vanilla LoRA-MoE) across all expert counts in both task performance and expert utilization, validated across unimodal and multimodal mathematical reasoning benchmarks.
- Our experiments provide the first empirical evidence that a scalar reward in RFT can align not only a model’s external behavior but also its internal mechanisms, opening new avenues for the principled alignment of complex model architectures.

## 2 RELATED WORK

**Modular and Mixture-of-Experts PEFT.** To enhance the capacity and versatility of PEFT, researchers have integrated Mixture-of-Experts (MoE) principles into LoRA, creating architectures like LoRA-MoE (Dou et al., 2024), MixLoRA (Li et al., 2024a), and MoCLE (Gou et al., 2024). These methods have shown effectiveness in reducing task interference and improving knowledge retention. Beyond adapterized MoE, modern routing builds on early conditional computation and deep MoE ideas (Cho & Bengio, 2014; Eigen et al., 2013) and on system-scale routed Transformers such as GShard and Switch (Lepikhin et al., 2020; Fedus et al., 2022). Most approaches optimize expert routing during supervised pretraining or SFT, typically using auxiliary load-balancing objectives to prevent expert collapse and under-utilization (Shazeer et al., 2017; Fedus et al., 2022). Alternative balancing mechanisms include the balanced assignment of BASE Layers (Lewis et al., 2021) and non-parametric routing via Hash Layers (Roller et al., 2021). Training stability and routing fluctuation have been studied extensively, with design guidelines in ST-MoE (Zoph et al., 2022), two-stage stabilized routing in StableMoE (Dai et al., 2022), and empirical analysis of expert-load dynamics (Cong et al., 2024a). However, these strategies remain predominantly designed for differentiable supervised training; integrating comparable mechanism-aware supervision into reinforcement fine-tuning is still underexplored.

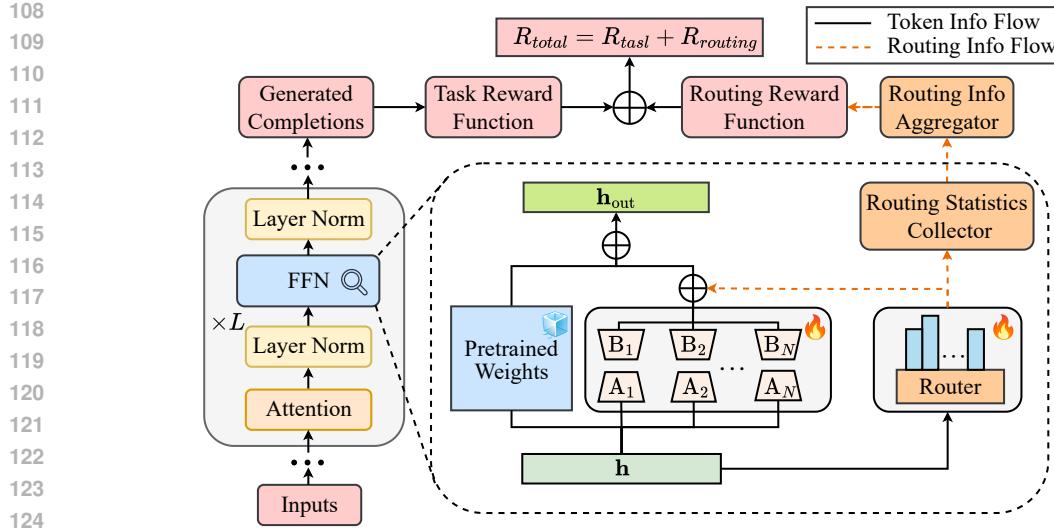


Figure 2: Overview of the RO-GRPO framework. A mechanism-aware reward  $R_{route}$  is computed from internal routing statistics and combined with the task reward  $R_{task}$ . The resulting unified reward  $R_{total}$  guides the GRPO update to jointly optimize task performance and routing efficiency.

**Alignment and Optimization of LLMs.** Reinforcement Learning from Human Feedback (RLHF) has become the dominant approach for aligning large language models with human intentions (Ouyang et al., 2022; Kaufmann et al., 2024). Proximal Policy Optimization (PPO) (Schulman et al., 2017) and its variants, such as Direct Preference Optimization (DPO) (Rafailov et al., 2024) and Group Relative Policy Optimization (GRPO) (Shao et al., 2024), have demonstrated strong performance on complex reasoning and instruction following. These RLHF methods typically optimize a scalar task-based reward and do not incorporate supervision for internal mechanisms such as expert routing. In MoE-based models, this limitation can lead to expert collapse or inefficient parameter utilization when using RLHF directly (Fedus et al., 2022; Harvey et al., 2025). While auxiliary losses have been used to encourage balanced routing in supervised settings, integrating such mechanism-aware supervision into reinforcement learning remains an open problem.

### 3 METHODOLOGY

In this section, we introduce **RO-GRPO** (Routing-Optimized Group Relative Policy Optimization), a framework designed to optimize the internal routing of LoRA-MoE models by incorporating mechanism-aware supervision into the reinforcement learning loop. We first review the preliminaries of GRPO and LoRA-MoE, then describe the core challenge of unguided routing in RLHF, and finally detail our proposed solution.

#### 3.1 PRELIMINARIES: GRPO AND LORA-MOE

**Group Relative Policy Optimization.** GRPO (Shao et al., 2024) is a critic-free RL algorithm that aligns an LLM policy  $\pi_\theta$  by maximizing the expected task reward over a dataset of prompts  $\mathcal{D}$ . Its objective is to maximize  $\mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(\cdot|x)} [R_{task}(y)]$ , where  $\pi_\theta$  is the LLM policy,  $\mathcal{D}$  is the dataset of prompts, and  $R_{task}$  is the scalar task reward evaluating the output  $y$ .

**LoRA-MoE Architecture.** A LoRA-MoE layer modifies the output of a frozen pretrained layer. It consists of a trainable router network and a set of  $E$  parallel LoRA experts,  $\{(\mathbf{A}_e, \mathbf{B}_e)\}_{e=1}^E$ . For an input token representation  $\mathbf{h}$ , the router first computes a gating probability vector  $\mathbf{p} = \text{softmax}(\mathbf{W}_r \mathbf{h})$ , where  $\mathbf{W}_r$  is the router's weight matrix. The final output of the layer is:

$$\mathbf{h}_{out} = \mathbf{h} + \left( \sum_{e=1}^E p(e | \mathbf{h}) \mathbf{B}_e \mathbf{A}_e \right) \mathbf{h}. \quad (1)$$

162 3.2 THE CHALLENGE: UNGUIDED MoE ROUTING  
163

164 A fundamental disconnect arises when a LoRA-MoE model is fine-tuned using an RLHF algorithm  
165 like GRPO. The optimization process is blind to the router’s decisions  $\mathbf{p}$ , as the task reward  $R_{\text{task}}$   
166 evaluates only the final output. This lack of explicit supervision leads to two well-documented  
167 failure modes in MoE training (Fedus et al., 2022):

- 168 • Expert Collapse: The router defaults to choosing a small subset of experts, leading to severe  
169 load imbalance and wasted parametric capacity.
- 170 • Routing Indecision: The router generates high-entropy distributions (i.e., low-confidence  
171 decisions), failing to foster expert specialization.

173 Our goal is to augment the RLHF objective with an internal, mechanism-aware reward signal that  
174 directly addresses these failure modes.

175 3.3 RO-GRPO: MECHANISM-AWARE REWARDS  
176

178 As depicted in Figure 2, our method collects internal routing statistics during policy generation.  
179 These statistics are used to compute  $R_{\text{route}}$ , which is then combined with the primary task reward,  
180  $R_{\text{task}}$ . This is achieved in three steps.

181 3.3.1 STEP 1: QUANTIFYING ROUTING EFFICIENCY.  
182

183 For each generated sample, we collect the routing probability vectors from all activated LoRA-MoE  
184 modules. Let  $M$  be the number of such modules in the model and  $T$  be the total number of tokens  
185 routed during generation. We quantify routing efficiency using two metrics aggregated from these  
186 statistics. First, we measure **routing confidence** using the mean Shannon entropy over all individual  
187 token routing decisions. A lower value indicates more decisive routing. For the multiset of all  $T$   
188 routing vectors  $\{\mathbf{p}_i\}_{i=1}^T$ , define the token-wise entropy as  $H(\mathbf{p}) := -\sum_{e=1}^E p_e \ln p_e$  (a small  $\epsilon$  is  
189 added in implementation). The average entropy is

$$190 \bar{H} = \frac{1}{T} \sum_{i=1}^T H(\mathbf{p}_i). \quad (2)$$

193 Second, we measure **load balance** by assessing expert utilization across the  $M$  LoRA-MoE  
194 modules. For each module  $m \in \{1, \dots, M\}$ , we first compute its average expert utilization vector  $\bar{\mathbf{p}}_m$   
195 by averaging the routing vectors of all tokens that pass through it. The final metric  $\bar{\mathcal{M}}$  is the mean  
196 of the Mean Squared Errors (MSE) calculated for each module relative to a uniform distribution  $\mathbf{u}$ :

$$198 \bar{\mathcal{M}} = \frac{1}{M} \sum_{m=1}^M \frac{1}{E} \left\| \bar{\mathbf{p}}_m - \frac{1}{E} \mathbf{1} \right\|_2^2. \quad (3)$$

200 For stable integration into the reward function, we normalize these metrics to an approximate  $[0, 1]$   
201 range, yielding  $\mathcal{H}_{\text{norm}} = \bar{H} / \ln E$ ,  $\mathcal{M}_{\text{norm}} = \bar{\mathcal{M}} / ((E - 1) / E^2)$ , for use in the reward calculation.  
202

203 3.3.2 STEP 2: FORMULATING THE ROUTING REWARD.  
204

205 We propose two distinct strategies to transform these metrics into a scalar reward  $R_{\text{route}}$ .

206 **RO-GRPO (Smooth): Curriculum-Based Reward Scheduling.** This strategy employs a curriculum  
207 that initially encourages confident routing (low entropy) and then transitions to promoting load  
208 balance (low MSE). As detailed in the Discussion section, this curriculum aligns the reward signal  
209 with the natural training dynamics of MoE models, defining the reward as follows:

$$210 R_{\text{route}} = -w_{\text{route}} (w_H(t) \cdot \mathcal{H}_{\text{norm}} + w_B(t) \cdot \mathcal{M}_{\text{norm}}), \quad (4)$$

212 where weights  $w_H(t)$  and  $w_B(t)$  are dynamically scheduled based on training progress  $t \in [0, 1]$   
213 using a sigmoid function  $\sigma(t) = \frac{1}{1+e^{-k(t-c)}}$  with steepness  $k$  and center  $c$ :

$$214 w_H(t) = \lambda_H^{\text{start}} \cdot (1 - \sigma(t)), \quad (5)$$

$$215 w_B(t) = \lambda_B^{\text{end}} \cdot \sigma(t). \quad (6)$$

216  
 217 Table 1: Performance on unimodal mathematical reasoning benchmarks. We compare task accuracy  
 218 (%) , trainable parameter count (#Param) , and internal routing metrics. **The Config column specifies**  
 219 **the adapter structure, denoted as rank  $r$  for LoRA or  $E \times r$  for LoRA-MoE.**

Unimodal Mathematical Reasoning (Qwen2.5-7B-Instruct on NuminaMath-TIR-2k)								
Method	Config	#Param	GSM8K	MATH	SVAMP	MGSM	Entropy	MSE
Base (zero-shot)	-	0	87.34	70.42	91.33	53.64	-	-
GRPO (LoRA)	16	30.3M	88.48	70.38	90.67	50.00	-	-
	32	<b>60.6M</b>	<b>90.45</b>	<b>69.54</b>	<b>93.00</b>	<b>47.10</b>	-	-
	64	<b>121.1M</b>	<b>90.14</b>	<b>49.02</b>	<b>92.67</b>	<b>53.67</b>	-	-
GRPO (LoRA-MoE)	2×8	31.7M	89.39	70.36	90.00	61.75	0.640	0.020
	4×8	63.4M	89.39	70.40	91.30	46.15	0.651	0.009
	8×8	126.9M	90.22	70.44	91.00	52.04	0.655	0.008
Aux-Loss (LoRA-MoE)	2×8	<b>31.7M</b>	<b>86.73</b>	<b>69.50</b>	<b>92.33</b>	<b>57.27</b>	<b>0.632</b>	<b>0.036</b>
	4×8	<b>63.4M</b>	<b>87.11</b>	<b>70.10</b>	<b>91.00</b>	<b>56.36</b>	<b>0.540</b>	<b>0.024</b>
	8×8	<b>126.9M</b>	<b>87.04</b>	<b>69.54</b>	<b>91.33</b>	<b>56.66</b>	<b>0.645</b>	<b>0.019</b>
RO-GRPO (Smooth)	2×8	31.7M	<b>91.51</b>	<b>70.64</b>	91.00	<b>62.18</b>	0.639	0.016
	4×8	63.4M	90.67	70.62	92.00	52.58	0.651	0.009
	8×8	126.9M	90.98	69.78	92.67	52.04	0.656	0.006
RO-GRPO (Relative)	2×8	31.7M	90.22	70.58	91.33	59.45	0.639	0.017
	4×8	63.4M	89.76	69.88	<b>93.33</b>	54.58	0.651	0.008
	8×8	126.9M	90.52	70.18	92.67	51.96	0.655	0.007

240 **RO-GRPO (Relative): Relative Improvement Gating.** This strategy provides a sparse, adaptive  
 241 reward based on a historical baseline, encouraging continuous self-improvement and avoiding the  
 242 need to manually balance the two routing objectives. A constant positive reward  $C$  is granted only  
 243 if both routing confidence and load balance improve simultaneously relative to their exponential  
 244 moving averages ( $\bar{\mathcal{H}}_{\text{hist}}$ ,  $\bar{\mathcal{M}}_{\text{hist}}$ ):

$$R_{\text{route}} = C \mathbf{1}\{\mathcal{H}_{\text{norm}} < \bar{\mathcal{H}}_{\text{hist}} \wedge \mathcal{M}_{\text{norm}} < \bar{\mathcal{M}}_{\text{hist}}\}. \quad (7)$$

### 246 3.3.3 STEP 3: UNIFIED OPTIMIZATION VIA POLICY GRADIENT.

248 The mechanism-aware reward  $R_{\text{route}}$  is combined with the external task reward to form the total  
 249 reward  $R_{\text{total}}(y) = R_{\text{task}}(y) + R_{\text{route}}(y)$ . The computation of  $R_{\text{route}}$  is non-differentiable; its  
 250 gradient is propagated implicitly through the policy gradient update. In the GRPO framework,  
 251 a group of responses  $\{y_i\}$  is sampled for each prompt and evaluated using  $R_{\text{total}}$ . The resulting  
 252 rewards are used to compute group-relative advantages,  $\hat{A}_i$ , which in turn guide the policy update.  
 253 The objective can be summarized as:

$$\mathcal{J}_{\text{RO-GRPO}}(\theta) \approx \mathbb{E}\left[\sum_i \log \pi_{\theta}(y_i|x) \hat{A}_i - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}})\right]. \quad (8)$$

256 By integrating  $R_{\text{route}}$  into the advantage calculation, RO-GRPO guides the policy  $\pi_{\theta}$  to generate  
 257 outputs that improve task performance while also exhibiting efficient routing, all without requiring  
 258 differentiable auxiliary losses.

## 260 4 EXPERIMENTS

262 We experimentally validate RO-GRPO by testing three core hypotheses: (1) applying LoRA-MoE  
 263 with GRPO leads to suboptimal routing and underutilized parameters; (2) our mechanism-aware  
 264 reward framework, RO-GRPO, mitigates these routing issues; and (3) these internal improvements  
 265 translate to better task performance.

### 267 4.1 EXPERIMENTAL SETUP

268 **Tasks and Datasets.** We evaluate RO-GRPO on challenging mathematical reasoning tasks. Per-  
 269 formance on these tasks relies on precise, multi-step deduction, making it sensitive to the model’s

270 Table 2: Performance on multimodal mathematical reasoning benchmarks. We follow the same  
 271 evaluation setup as in the unimodal experiments.

Multimodal Mathematical Reasoning (Qwen2.5-VL-7B-Instruct on Geometry3k)									
Method	Config	#Param	Geo3k	MathVista	MathVerse	WeMath	Entropy	MSE	
Base (zero-shot)	-	0	37.44	46.50	26.50	56.95	-	-	
GRPO (LoRA)	16	30.3M	38.44	58.60	<b>33.30</b>	63.97	-	-	
	32	<b>60.6M</b>	<b>38.10</b>	<b>59.30</b>	<b>23.22</b>	<b>53.85</b>	-	-	
	64	<b>121.1M</b>	<b>33.28</b>	<b>55.90</b>	<b>25.43</b>	<b>53.91</b>	-	-	
GRPO (LoRA-MoE)	2×8	31.7M	38.27	57.90	30.99	63.74	0.619	0.038	
	4×8	63.4M	28.95	56.40	30.30	63.45	0.637	0.019	
	8×8	126.9M	33.11	55.00	31.78	61.49	0.649	0.012	
Aux-Loss (LoRA-MoE)	2×8	<b>31.7M</b>	<b>39.60</b>	<b>56.20</b>	<b>30.03</b>	<b>62.81</b>	<b>0.621</b>	<b>0.060</b>	
	4×8	<b>63.4M</b>	<b>41.43</b>	<b>60.50</b>	<b>27.23</b>	<b>62.87</b>	<b>0.634</b>	<b>0.036</b>	
	8×8	<b>126.9M</b>	<b>40.43</b>	<b>54.40</b>	<b>32.13</b>	<b>65.80</b>	<b>0.649</b>	<b>0.019</b>	
RO-GRPO (Smooth)	2×8	31.7M	38.94	58.70	30.48	66.09	0.630	0.033	
	4×8	63.4M	40.10	58.30	28.73	64.14	0.642	0.014	
	8×8	126.9M	38.94	58.90	27.89	64.10	0.648	0.011	
RO-GRPO (Relative)	2×8	31.7M	<b>41.93</b>	55.80	<b>33.30</b>	60.98	0.624	0.036	
	4×8	63.4M	40.27	60.20	31.29	<b>66.26</b>	0.645	0.013	
	8×8	126.9M	40.16	60.10	32.03	63.97	0.636	0.010	

291  
 292 expert utilization and thus an ideal testbed for our approach. To demonstrate the versatility of our  
 293 method, we conduct experiments in both unimodal and multimodal settings.

294  
 295 For unimodal experiments, we fine-tune on NuminaMath-TIR (Li et al., 2024b) and evaluate on the  
 296 established benchmarks GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), SVAMP (Pa-  
 297 tel et al., 2021), and MGSM (Shi et al., 2022). For multimodal experiments, we fine-tune on Ge-  
 298 ometry3k (Lu et al., 2021) and evaluate on its test set, alongside MathVista (Lu et al., 2024), Math-  
 299 Verse (Zhang et al., 2024), and WeMath (Qiao et al., 2024).

300  
 301 **Models and Baselines.** We use the open-source Qwen2.5-7B-Instruct (Qwen et al., 2025) and  
 302 Qwen2.5-VL-7B-Instruct (Bai et al., 2025) models, chosen for their strong foundational perfor-  
 303 mance in mathematical reasoning. We compare five configurations: (1) **Base**, the original pretrained  
 304 model evaluated in a zero-shot setting; (2) **GRPO (LoRA)**, a standard LoRA baseline fine-tuned  
 305 with GRPO representing a typical PEFT approach; (3) **GRPO (LoRA-MoE)**, a LoRA-MoE model  
 306 trained with GRPO using only the task reward to isolate the effect of unguided routing; (4) **Aux-  
 307 Loss (LoRA-MoE)**, a baseline where routing objectives are added as auxiliary losses to the GRPO  
 308 objective, using the same scheduling as the Smooth strategy; (5) **RO-GRPO (Smooth)**, our method  
 309 with the curriculum-based reward scheduling strategy (Section 3.3); and (6) **RO-GRPO (Relative)**,  
 310 our method with the relative improvement gating strategy for the routing reward (Section 3.3).

311  
 312 **Evaluation Metrics.** We evaluate both task performance and internal mechanism efficiency. Task  
 313 Performance is measured by accuracy (%) on the respective benchmarks. Routing Performance is  
 314 quantified by two metrics: (1) **routing entropy**, the average per-token Shannon entropy as formulated  
 315 in Eq. (2), indicating decision confidence; and (2) **load balancing MSE**, the mean squared error  
 316 between the expert utilization distribution and a uniform one as formulated in Eq. (3), indicating  
 317 load balance. We report these raw, un-normalized values for direct interpretability.

318  
 319 **Implementation Details.** To ensure a fair comparison, we control for trainable-parameter budget:  
 320 the LoRA baseline uses a rank of 16, while LoRA-MoE models use  $E$  experts ( $E \in \{2, 4, 8\}$ ), each  
 321 with rank of 8. For LoRA-MoE, modules are inserted into the gate, up, and down projections  
 322 of the Feed-Forward Network (FFN) in each transformer block. During training, only the PEFT  
 323 parameters are updated while the base model weights remain frozen. Across all experiments, we  
 324 use a consistent system prompt for both training and evaluation to encourage step-by-step reasoning.  
 325 The overall routing reward  $R_{\text{route}}$  is integrated into the total reward using a global scaling coefficient  
 326 of  $w_{\text{route}} = 0.2$ . Key hyperparameters for our routing strategies were determined through validation.

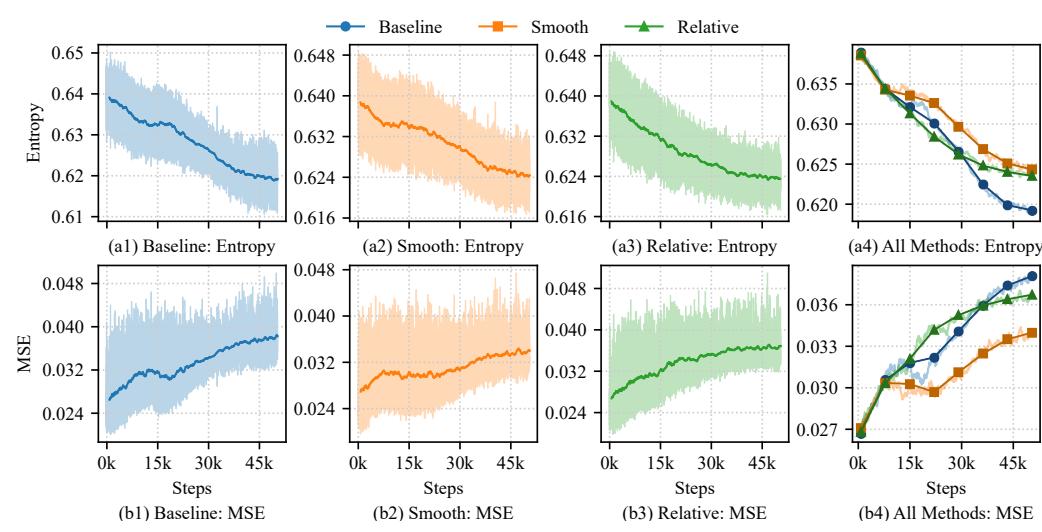


Figure 3: Training dynamics of routing metrics on the unimodal mathematical reasoning task. (Top) Average routing entropy over the course of training. (Bottom) Load balancing MSE over the course of training.

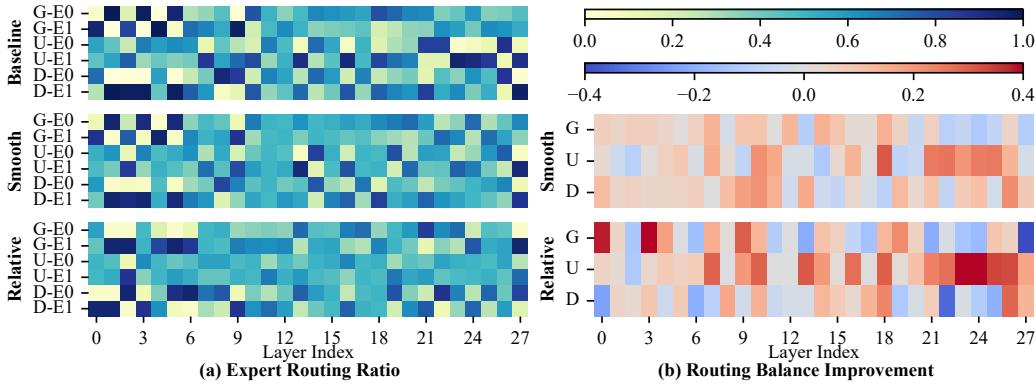


Figure 4: Visual analysis of routing behavior improvements with RO-GRPO on the MathVista benchmark. (a) **Left Panel:** Heatmaps show the routing ratio of the most frequently selected expert for the baseline and our two RO-GRPO methods. Darker colors represent a higher selection ratio for the dominant expert. (b) **Right Panel:** Heatmaps quantify the improvement in routing balance relative to the vanilla LoRA-MoE baseline. Positive values (warmer colors) indicate a reduced routing ratio for the dominant expert, signifying a shift toward the desired  $1/E$  equilibrium.

For the Smooth strategy, we set the initial entropy weight  $\lambda_H^{\text{start}} = 0.5$  and the final balance weight  $\lambda_B^{\text{end}} = 2.0$ . For the Relative strategy, the performance baseline was calculated over a moving window of the 1000 most recent samples. Further details are in the Appendix.

## 4.2 MAIN RESULTS

As shown in Tables 1 and 2, RO-GRPO yields consistent performance gains over vanilla GRPO with LoRA-MoE across all expert counts ( $E \in \{2, 4, 8\}$ ) in both unimodal and multimodal settings. The two reward variants show complementary strengths: the Smooth strategy performs best on GSM8K and SVAMP, while the Relative strategy excels on Geometry3k, MathVista, and WeMath.

**Unimodal.** On GSM8K, RO-GRPO (Smooth,  $E=2$ ) achieves the top score of 91.51%, an improvement of **+1.37** pp over GRPO (LoRA) and **+1.29** pp over the best-performing vanilla LoRA-MoE model (at  $E=8$ ). On SVAMP, RO-GRPO (Relative,  $E=4$ ) obtains 93.33%, surpassing GRPO (LoRA) by **+0.33** pp and vanilla LoRA-MoE (at  $E=4$ ) by **+2.03** pp. While improvements on MATH

378 are modest, they are consistent at matched expert counts (e.g., 70.64% vs. 70.36% for  $E=2$  Smooth  
 379 vs. vanilla). The largest gain on MGSM is at  $E=4$ , where RO-GRPO (Relative) achieves 54.58%,  
 380 an **+8.43** pp increase over the vanilla model.  
 381

**Multimodal.** RO-GRPO (Relative,  $E=2$ ) attains the highest Geometry3k score of 41.93%, outperforming vanilla LoRA-MoE ( $E=2$ ) by **+0.5** pp and GRPO (LoRA) by **+3.49** pp. On WeMath, the best results are with RO-GRPO (Relative,  $E=4$ ), which reaches 66.26%, gains of **+2.29** pp over GRPO (LoRA), and **+2.52** pp over the best vanilla LoRA-MoE model (at  $E=2$ ), respectively. On MathVerse, the top performance of 33.30% is shared by GRPO (LoRA) and RO-GRPO (Relative,  $E=2$ ).  
 382  
 383  
 384  
 385  
 386  
 387

Overall, across expert sizes, at matched  $E$  our routing-aware training either matches or surpasses vanilla LoRA-MoE on nearly every benchmark, with the largest margins on Geometry3k and WeMath. These results reaffirm that aligning the router with mechanism-aware rewards translates into stronger task performance.  
 388  
 389  
 390  
 391

### 392 4.3 ANALYSIS OF ROUTING MECHANISM 393

**394 Unguided routing under vanilla GRPO is brittle.** At  $E=2$  in the multimodal setting, vanilla  
 395 LoRA-MoE appears confident (Entropy 0.619) but exhibits routing collapse (MSE 0.038). At larger  
 396  $E$ , the raw MSE decreases (e.g., multimodal 0.019 and 0.012 for  $E=4$  and  $E=8$ ), yet accuracy does  
 397 not reliably improve and can even drop (Geometry3k score of 28.95% at  $E=4$ ), revealing instability  
 398 in the absence of mechanism-aware feedback.  
 399

**400 RO-GRPO restores balance at matched  $E$ .** Across all experimental configurations, RO-GRPO  
 401 reduces or matches the MSE of the vanilla baseline at the same  $E$  (unimodal: 0.020 → 0.016/0.017,  
 402 0.009 → 0.009/0.008, 0.008 → 0.006/0.007; multimodal: 0.038 → 0.033/0.036, 0.019 →  
 403 0.014/0.013, 0.012 → 0.011/0.010), while maintaining comparable entropy. These improvements  
 404 in routing correspond to the largest accuracy gains on Geometry3k, WeMath, and SVAMP.  
 405

**406 Routing rewards mitigate text degeneration.** On Geometry3k with  $E=4$ , the vanilla GRPO  
 407 (LoRA-MoE) model exhibits repetitive-loop failures in 7.5% of generations and scores 28.95%. In  
 408 contrast, RO-GRPO (Smooth) reduces these failures to 0.17% and RO-GRPO (Relative) eliminates  
 409 them entirely, achieving accuracies of 40.10% and 40.27%, respectively.  
 410

**411 Mechanism-Aware Rewards outperform Auxiliary Losses.** As shown in Tables 1 and 2, the  
 412 Aux-Loss baseline consistently underperforms RO-GRPO, particularly on unimodal tasks. While  
 413 auxiliary losses successfully reduce routing entropy, they fail to improve load balance, often yielding  
 414 higher MSE values compared to our reward-based approach (e.g., 0.036 vs. 0.016 on GSM8K with  
 415  $E=2$ ). Furthermore, the auxiliary-loss method tends to generate significantly longer sequences  
 416 without yielding commensurate accuracy gains, as detailed in Appendix H and Appendix D.  
 417

### 418 4.4 ABLATION AND CAUSAL VERIFICATION 419

Ablation experiments on GSM8K confirm our approach’s integrity (Table 3).  
 420

**421 Contribution of Reward Components.** We first investigate the individual contributions of the  
 422 confidence ( $R_H$ ) and balancing ( $R_B$ ) rewards. As shown in Table 3, removing either component  
 423 from our best-performing model, RO-GRPO (Smooth), reduces accuracy. Specifically, removing  
 424 the balancing reward (w/o  $R_B$ ) or the confidence reward (w/o  $R_H$ ) reduces the GSM8K score  
 425 from 91.51% to 90.75% and 89.92%, respectively. This dependency is more pronounced for the RO-  
 426 GRPO (Relative) variant: removing its balancing reward (w/o  $R_B$ ) causes performance to drop to  
 427 89.01%, below the vanilla baseline. These results demonstrate that the two reward components are  
 428 synergistic and critical for optimal performance.  
 429

**430 Contribution of the Reward Signal.** To ensure performance gains are driven by meaningful feed-  
 431 back, we performed a control experiment. In the Shuffled Control, we randomly permuted routing  
 432 rewards within each batch, breaking the causal link between an action and its reward. As shown in  
 433 Table 3, performance under this condition dropped significantly for both the Smooth (89.23%) and  
 434

432  
 433 Table 3: Ablation and causal analysis on  
 434 GSM8K ( $E=2$ ). Our full RO-GRPO model sig-  
 435 nificantly outperforms the vanilla baseline. Sub-  
 436 sequent experiments demonstrate that both re-  
 437 ward components ( $R_H, R_B$ ) are necessary for  
 438 optimal performance, and control experiments  
 439 validate that the gains are causally driven by our  
 440 targeted reward signal.

Configuration	GSM8K	Entropy	MSE
RO-GRPO (Smooth)	91.51	0.639	0.016
RO-GRPO (Relative)	90.22	0.639	0.017
GRPO (LoRA-MoE)	89.39	0.640	0.020
<i>Ablations on Reward Components:</i>			
w/o $R_B$ (Smooth)	90.75	0.639	0.018
w/o $R_H$ (Smooth)	89.92	0.639	0.019
w/o $R_B$ (Relative)	89.01	0.638	0.019
w/o $R_H$ (Relative)	90.14	0.637	0.019
<i>Causal &amp; Sanity Controls:</i>			
Shuffled (Smooth)	89.23	0.640	0.018
Shuffled (Relative)	89.28	0.641	0.020

451  
 452  
 453  
 454 Relative (89.28%) variants, falling to the level of the vanilla GRPO (LoRA-MoE) baseline (89.39%).  
 455 This result strongly suggests the gains from RO-GRPO are causally driven by the targeted feedback  
 456 from our reward signal, not by an artifact of the reward structure.  
 457

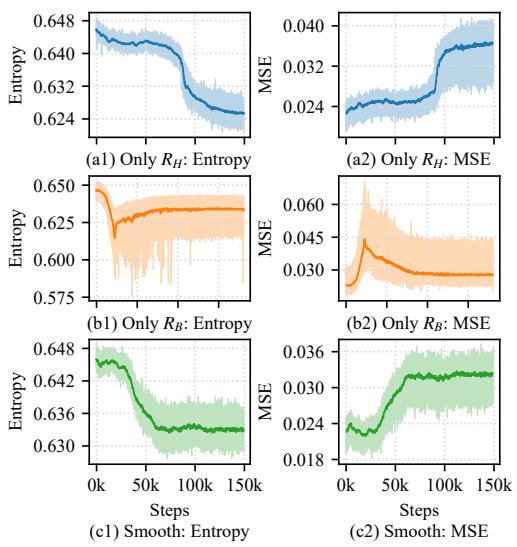
## 5 DISCUSSION

460 Our experiments demonstrate the empirical success of RO-GRPO across  $E=2, 4, 8$ . This section  
 461 analyzes why a unified scalar reward can supervise a model’s internal router and why this becomes  
 462 more important as  $E$  grows. A more detailed derivation is available in Appendix G.  
 463

464 **The Rationale for a Curriculum-Based Reward.** The Smooth curriculum is effective because  
 465 single-objective optimization is suboptimal: rewarding only low entropy degrades balance (MSE  
 466 rises), whereas rewarding only balance is initially too weak to shape specialization (Figure 5). By  
 467 first encouraging confident routing and then increasing pressure on balance, the curriculum builds  
 468 specialized experts and subsequently organizes them. This dynamic mirrors our empirical trends  
 469 at  $E=4, 8$ , where mechanism-aware supervision not only improves accuracy but also suppresses  
 470 degeneration on Geometry3k.

471 **Rewards vs. Auxiliary Losses in RL.** A critical insight from our study is the superiority of  
 472 mechanism-aware rewards over auxiliary losses in the GRPO framework. When routing supervi-  
 473 sion is formulated as a reward, it is integrated into the group-relative advantage calculation. This allows  
 474 the model to learn trade-offs: a trajectory with slightly imbalanced routing can still receive a positive  
 475 advantage if it yields a correct answer. Conversely, an auxiliary loss applies a uniform penalty to all  
 476 trajectories in a batch regardless of their task success. This rigid penalization can suppress useful but  
 477 unconventional routing patterns required for complex tasks, leading to the suboptimal performance.  
 478 Additionally, our approach requires no extra gradient backpropagation or VRAM overhead.  
 479

480 **Grounding the Reward Components.** The confidence reward,  $R_H$ , which promotes low-entropy  
 481 routing, can be understood through the Information Bottleneck (IB) principle (Tishby et al., 2000).  
 482 The IB principle states that an optimal representation should compress an input while preserving  
 483 task-relevant information. In our framework, the router’s decision acts as this bottleneck. By re-  
 484 warding low-entropy (confident) decisions, RO-GRPO incentivizes the router to learn a minimal  
 485 sufficient representation of its input. It is encouraged to discard noisy features and focus on infor-  
 486 mation predictive of task success, a process that naturally fosters expert specialization.



487  
 488 Figure 5: Training dynamics of routing metrics  
 489 when the  $R_{\text{task}}$  is set to zero.

486 The balancing reward  $R_B$  directly optimizes parameter utilization. Maximizing our balancing  
 487 reward, which is formulated using MSE, is formally equivalent to minimizing the variance of the  
 488 expert load distribution. This ensures the reward signal provides a direct and efficient gradient for  
 489 combating routing collapse and ensuring the model leverages its full parametric capacity, a principle  
 490 established in supervised MoE training (Shazeer et al., 2017).

## 492 6 CONCLUSION

494 We addressed a core limitation of applying LoRA-MoE to GRPO: the task reward is blind to routing.  
 495 RO-GRPO remedies this by transforming routing statistics into a mechanism-aware reward that  
 496 plugs into GRPO without architecture changes or extra stages. Across  $E=2, 4, 8$  and both unimodal  
 497 and multimodal math reasoning, RO-GRPO improves load balance at matched  $E$ , boosts accuracy,  
 498 and reduces text degeneration. These results indicate that reinforcement learning can align not only  
 499 external behavior but also internal mechanisms, suggesting a path toward principled alignment for  
 500 complex modular architectures.

## 502 REPRODUCIBILITY STATEMENT

504 To ensure the reproducibility of our findings, we provide a comprehensive set of resources. The com-  
 505 plete source code for all experiments, including model implementation and training and evaluation  
 506 scripts, is available in the supplementary material. Further details on the experimental environment,  
 507 including the specific hardware and software configurations used, are documented in Appendix A.  
 508 A complete list of hyperparameters for all model configurations and our routing reward strategies is  
 509 provided in Appendix B, alongside pseudocode for our reward calculation algorithms.

## 511 REFERENCES

513 Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang,  
 514 Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan,  
 515 Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng,  
 516 Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-v1 technical report, 2025.  
 517 URL <https://arxiv.org/abs/2502.13923>.

518 Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. Towards understanding mixture  
 519 of experts in deep learning, 2022. URL <https://arxiv.org/abs/2208.02813>.

520 Kyunghyun Cho and Yoshua Bengio. Exponentially increasing the capacity-to-computation ratio  
 521 for conditional computation in deep learning. *arXiv preprint arXiv:1406.7362*, 2014.

523 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,  
 524 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John  
 525 Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*,  
 526 2021.

527 Peizhuang Cong, Aomufei Yuan, Shimao Chen, Yuxuan Tian, Bowen Ye, and Tong Yang. Prediction  
 528 is all moe needs: Expert load distribution goes from fluctuating to stabilizing. *arXiv preprint  
 529 arXiv:2404.16914*, 2024a.

531 Peizhuang Cong, Aomufei Yuan, Shimao Chen, Yuxuan Tian, Bowen Ye, and Tong Yang. Prediction  
 532 is all moe needs: Expert load distribution goes from fluctuating to stabilizing, 2024b. URL  
 533 <https://arxiv.org/abs/2404.16914>.

534 Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. Stable-  
 535 moe: Stable routing strategy for mixture of experts. *arXiv preprint arXiv:2204.08396*, 2022.

536 Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi,  
 537 Xiao Wang, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing  
 538 Huang. Loramoe: Alleviate world knowledge forgetting in large language models via moe-style  
 539 plugin, 2024. URL <https://arxiv.org/abs/2312.09979>.

540 David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep  
 541 mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

542

543 William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter  
 544 models with simple and efficient sparsity, 2022. URL <https://arxiv.org/abs/2101.03961>.

545

546 Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T.  
 547 Kwok, and Yu Zhang. Mixture of cluster-conditional lora experts for vision-language instruction  
 548 tuning, 2024. URL <https://arxiv.org/abs/2312.12379>.

549

550 Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning  
 551 for large models: A comprehensive survey, 2024. URL <https://arxiv.org/abs/2403.14608>.

552

553 Daniel Fidel Harvey, George Weale, and Berk Yilmaz. Optimizing moe routers: Design, imple-  
 554 mentation, and evaluation in transformer models, 2025. URL <https://arxiv.org/abs/2506.16419>.

555

556 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,  
 557 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021.  
 558 URL <https://arxiv.org/abs/2103.03874>.

559

560 Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,  
 561 and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL <https://arxiv.org/abs/2106.09685>.

562

563 Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement  
 564 learning from human feedback, 2024. URL <https://arxiv.org/abs/2312.14925>.

565

566 Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang,  
 567 Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with  
 568 conditional computation and automatic sharding, 2020. URL <https://arxiv.org/abs/2006.16668>.

569

570 Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers:  
 571 Simplifying training of large, sparse models. In *International Conference on Machine Learning*,  
 572 pp. 6265–6274. PMLR, 2021.

573

574 Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan  
 575 Zhang, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. Mixlora: Enhancing large language  
 576 models fine-tuning with lora-based mixture of experts, 2024a. URL <https://arxiv.org/abs/2404.15159>.

577

578 Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif  
 579 Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in  
 580 ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*,  
 581 13(9):9, 2024b.

582

583 Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu.  
 584 Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning,  
 585 2021. URL <https://arxiv.org/abs/2105.04165>.

586

587 Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-  
 588 Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning  
 589 of foundation models in visual contexts, 2024. URL <https://arxiv.org/abs/2310.02255>.

590

591 Siyuan Mu and Sen Lin. A comprehensive survey of mixture-of-experts: Algorithms, theory, and  
 592 applications, 2025. URL <https://arxiv.org/abs/2503.07137>.

593

Nabil Omi, Siddhartha Sen, and Ali Farhadi. Load balancing mixture of experts with similarity  
 preserving routers, 2025. URL <https://arxiv.org/abs/2506.14038>.

594 Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong  
 595 Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kel-  
 596 ton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike,  
 597 and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.  
 598 URL <https://arxiv.org/abs/2203.02155>.

599  
 600 Arkil Patel, Satwik Bhattacharya, and Navin Goyal. Are nlp models really able to solve simple math  
 601 word problems?, 2021. URL <https://arxiv.org/abs/2103.07191>.

602 Runqi Qiao, Qiuna Tan, Guanting Dong, Minhui Wu, Chong Sun, Xiaoshuai Song, Zhuoma  
 603 GongQue, Shanglin Lei, Zhe Wei, MiaoXuan Zhang, Runfeng Qiao, Yifan Zhang, Xiao Zong,  
 604 Yida Xu, Muxi Diao, Zhimin Bao, Chen Li, and Honggang Zhang. We-math: Does your  
 605 large multimodal model achieve human-like mathematical reasoning?, 2024. URL <https://arxiv.org/abs/2407.01284>.

606  
 607 Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan  
 608 Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang,  
 609 Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin  
 610 Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,  
 611 Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,  
 612 Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.  
 613 URL <https://arxiv.org/abs/2412.15115>.

614  
 615 Rafael Raffailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and  
 616 Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model,  
 617 2024. URL <https://arxiv.org/abs/2305.18290>.

618  
 619 Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models.  
 620 *advances in neural information processing systems*, 34:17555–17566, 2021.

621  
 622 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy  
 623 optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.

624  
 625 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,  
 626 Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathe-  
 627 matical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.

628  
 629 Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton,  
 630 and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer.  
 631 *arXiv preprint arXiv:1701.06538*, 2017.

632  
 633 Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi,  
 634 Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Lan-  
 635 guage models are multilingual chain-of-thought reasoners, 2022. URL <https://arxiv.org/abs/2210.03057>.

636  
 637 Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method, 2000.  
 638 URL <https://arxiv.org/abs/physics/0004057>.

639  
 640 Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load  
 641 balancing strategy for mixture-of-experts, 2024a. URL <https://arxiv.org/abs/2408.15664>.

642  
 643 Xujia Wang, Haiyan Zhao, Shuo Wang, Hanqing Wang, and Zhiyuan Liu. Malora: Mixture  
 644 of asymmetric low-rank adaptation for enhanced multi-task learning, 2024b. URL <https://arxiv.org/abs/2410.22782>.

645  
 646 Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts, 2024. URL <https://arxiv.org/abs/2404.13628>.

648 Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou,  
649 Pan Lu, Kai-Wei Chang, Peng Gao, and Hongsheng Li. Mathverse: Does your multi-modal llm  
650 truly see the diagrams in visual math problems?, 2024. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2403.14624)  
651 2403.14624.

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

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702 **A COMPUTING INFRASTRUCTURE AND SOFTWARE**  
703704 All experiments were conducted on a high-performance computing cluster. The specific hardware  
705 and software configurations are provided to ensure full reproducibility.  
706707 **Hardware:** Each experiment was run on a single node equipped with 8x NVIDIA A800 (80GB  
708 VRAM) GPUs. Each node was powered by an Intel(R) Xeon(R) Platinum 8336C CPU with 1875  
709 GB of system RAM.  
710711 **Software:** The operating system was Ubuntu 20.04.6 LTS. The core software stack included:  
712713

- 714 Python 3.10.18
- 715 PyTorch 2.5.1 (built with CUDA 12.1)
- 716 CUDA Toolkit 12.2
- 717 Hugging Face Transformers 4.51.0
- 718 Hugging Face PEFT 0.14.0
- 719 The `ms-swift` framework, version 3.7.0.dev0, was used for all training scripts.

720 **B HYPERPARAMETER AND IMPLEMENTATION DETAILS**  
721722 Our approach to hyperparameter selection is  
723 designed to be both systematic and efficient.  
724 For established components of the training  
725 pipeline, such as the GRPO algorithm and the  
726 LoRA architecture, we adopted values from  
727 seminal works and common practices to estab-  
728 lish strong, competitive baselines. Our primary  
729 tuning efforts were concentrated on the novel  
730 parameters introduced by the RO-GRPO frame-  
731 work, ensuring a rigorous evaluation of our core  
732 contributions.  
733734 **Core Training and Architecture Parameters.**  
735 For all experiments, we used a learning rate of  
736  $1 \times 10^{-5}$  and a batch size of 64. The GRPO con-  
737 figuration included a KL coefficient ( $\beta$ ) of 0.1  
738 and sampling 8 responses per prompt ( $k = 8$ )  
739 for advantage estimation. For the base LoRA  
740 architecture, we set the rank to  $r = 16$  and al-  
741 pha to  $\alpha = 32$ . For our LoRA-MoE models, we  
742 used  $E \in \{2, 4, 8\}$  experts, each with a rank of  
743  $r = 8$  and an alpha of  $\alpha = 32$ , maintaining a  
744 similar parameter budget. The training duration  
745 was set to 1 epoch for the unimodal Numina-  
746 Math dataset and 3 epochs for the more com-  
747 plex multimodal Geometry3k dataset to ensure  
748 convergence. The external task reward weight  
749 was consistently set to 1.0.  
750751 **RO-GRPO Routing Reward Parameters.**  
752 The most critical hyperparameters are those  
753 governing the mechanism-aware routing re-  
754 ward,  $R_{\text{route}}$ . We conducted a grid search to de-  
755 termine the optimal settings for both our adap-  
756 tive strategies, using a held-out validation set.  
757758 For the **Curriculum-Based Reward Scheduling (Smooth)** strategy, we explored the key parame-  
759 ters controlling the curriculum’s shape and intensity. The search space included the final load bal-  
760761 Table 4: Hyperparameters for all experiments.  
762

Parameter	Value
<b>GRPO Configuration</b>	
Learning Rate	$1 \times 10^{-5}$
KL Coefficient ( $\beta$ )	0.1
Batch Size	64
Generations per Prompt ( $k$ )	8
Epochs (Unimodal)	1
Epochs (Multimodal)	3
<b>LoRA / LoRA-MoE Configuration</b>	
LoRA Rank ( $r$ )	16
LoRA-MoE Rank ( $r$ )	8 (per expert)
Number of Experts ( $E$ )	{2,4,8}
LoRA Alpha ( $\alpha$ )	32
LoRA Dropout	0.05
<b>RO-GRPO Specific (Optimal Values)</b>	
Global Routing Weight ( $w_{\text{route}}$ )	0.2
<i>Smooth Strategy</i>	
$\lambda_H^{\text{start}}$ (Entropy Weight Start)	0.5
$\lambda_B^{\text{end}}$ (Balance Weight End)	2.0
Sigmoid Steepness ( $k$ )	20.0
Sigmoid Center ( $c$ )	0.5
<i>Relative Strategy</i>	
History Window Size ( $S_{\text{hist}}$ )	1000
Reward Constant ( $C$ )	1.0

763 Note: Sigmoid steepness ( $k$ ) controls the transi-  
764 tion speed of the curriculum, and the center ( $c$ )  
765 defines the transition point in terms of training  
766 progress.  
767

756 ancing weight  $\lambda_B^{\text{end}} \in \{1.0, 2.0, 5.0\}$ , the initial entropy weight  $\lambda_H^{\text{start}} \in \{0.5, 1.0\}$ , and the sigmoid  
 757 steepness  $k \in \{15, 20, 25\}$ . The pseudocode for this strategy is detailed in Algorithm 2.  
 758

759 For the **Relative Improvement Gating (Relative)** strategy, the key parameter is the  
 760 `history_size`, which defines the window for the moving average baseline. We searched over  
 761 values in  $\{100, 500, 1000\}$ . The logic for this strategy is presented in Algorithm 1.

762 A separate grid search was performed for the global routing reward weight, which scales the entire  
 763  $R_{\text{route}}$  term, across the range  $\{0.1, 0.2, 0.5\}$ . Our experiments indicated that a weight of **0.2** provided  
 764 the best trade-off between improving task accuracy and optimizing routing efficiency (i.e., minimiz-  
 765 ing load balancing MSE and routing entropy). This value was used for all reported RO-GRPO  
 766 results.

767 The final, optimal hyperparameters selected through this process are summarized in Table 4.  
 768  
 769

770 **Algorithm 1** RO-GRPO Reward Calculation  
 771 (Relative Strategy)

772 **Input:** Routing statistics `stats`, history buffer  
 773  $B_{\text{hist}}$   
 774 **Parameters:** Reward constant  $C$ , history buffer  
 775 size  $S_{\text{hist}}$   
 776  
 777 {Require sufficient history to establish a baseline}  
 778  
 779 **if**  $|B_{\text{hist}}| < S_{\text{hist}}$  **then**  
 780     **return** 0  
 781 **end if**  
 782  
 783 {Compute metrics for the current sample}  
 784  $(\mathcal{M}_{\text{curr}}, \bar{\mathcal{H}}_{\text{curr}}) \leftarrow \text{ComputeMetrics}(\text{stats})$   
 785  
 786 {Compute historical average baseline}  
 787  $\mathcal{M}_{\text{hist}} \leftarrow \text{Average}(B_{\text{hist}}.\text{mse})$   
 788  $\bar{\mathcal{H}}_{\text{hist}} \leftarrow \text{Average}(B_{\text{hist}}.\text{entropy})$   
 789  
 790 {Grant reward only if both metrics improve}  
 791 **if**  $\mathcal{M}_{\text{curr}} < \mathcal{M}_{\text{hist}}$  **and**  $\bar{\mathcal{H}}_{\text{curr}} < \bar{\mathcal{H}}_{\text{hist}}$  **then**  
 792      $R_{\text{route}} \leftarrow C$   
 793 **else**  
 794      $R_{\text{route}} \leftarrow 0$   
 795 **end if**  
 796  
 797 {Update the history buffer with current metrics}  
 798  $\text{Update}(B_{\text{hist}}, (\mathcal{M}_{\text{curr}}, \bar{\mathcal{H}}_{\text{curr}}))$   
 799  
 800 **return**  $R_{\text{route}} = 0$

508 **Algorithm 2** RO-GRPO Reward Calculation  
 509 (Smooth Strategy)

510 **Input:** Routing statistics `stats`, current step  $t_{\text{curr}}$ ,  
 511 max steps  $t_{\text{max}}$   
 512 **Parameters:** Global weight  $w_{\text{route}}$ , entropy start  
 513 weight  $\lambda_H^{\text{start}}$ , balance end weight  $\lambda_B^{\text{end}}$ , sigmoid cen-  
 514 ter  $c$ , sigmoid steepness  $k$   
 515  
 516 {Calculate curriculum progress and sigmoid  
 517 value}  
 518  $p \leftarrow t_{\text{curr}}/t_{\text{max}}$   
 519  $\sigma \leftarrow (1 + e^{-k(p-c)})^{-1}$   
 520  
 521 {Schedule the weights for entropy and balance}  
 522  $w_H \leftarrow \lambda_H^{\text{start}} \cdot (1 - \sigma)$   
 523  $w_B \leftarrow \lambda_B^{\text{end}} \cdot \sigma$   
 524  
 525 {Compute average normalized metrics from stats}  
 526  
 527  $\bar{\mathcal{H}}_{\text{norm}} \leftarrow \text{AverageNormalizedEntropy}(\text{stats})$   
 528  $\mathcal{M}_{\text{norm}} \leftarrow \text{AverageNormalizedMSE}(\text{stats})$   
 529  
 530 {Calculate final reward (a negative penalty)}  
 531  $R_{\text{route}} \leftarrow -w_{\text{route}} \cdot (w_H \cdot \bar{\mathcal{H}}_{\text{norm}} + w_B \cdot \mathcal{M}_{\text{norm}})$   
 532  
 533 **return**  $R_{\text{route}} = 0$

799 **C SYSTEM PROMPT**

800  
 801 We used a consistent system prompt for all models during both training and evaluation to encourage  
 802 step-by-step reasoning. The prompt is provided below:  
 803

804  
 805 A conversation between User and Assistant. The user asks a question,  
 806 and the Assistant solves it. The assistant first thinks about the  
 807 reasoning process in the mind and then provides the user with the  
 808 answer. The reasoning process and answer are enclosed within `<think>`  
 809 `</think>` and `<answer>` `</answer>` tags, respectively, i.e., `<think>`  
 reasoning process here`</think>``<answer>` answer here `</answer>`

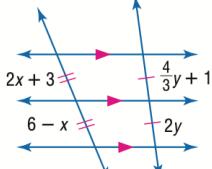
810 **D QUALITATIVE CASE STUDY**

811  
 812 To provide a comprehensive qualitative analysis of the impact of our method, we present case studies  
 813 spanning both unimodal arithmetic reasoning (GSM8K) and multimodal geometric reasoning (Ge-  
 814 ometry3K). Tables 5 and Table 6 contrast the specific generation trajectories of the baseline meth-  
 815 ods against our RO-GRPO framework. These examples empirically demonstrate that RO-GRPO not  
 816 only corrects the logical hallucinations and repetitive loops observed in baselines to ensure higher  
 817 accuracy, but also achieves these correct solutions with reduced token consumption, thereby improv-  
 818 ing inference efficiency.

819 Table 5: Case study comparing method outputs on a sample from the GSM8K benchmark.

821 Aspect	822 Content
823 <b>Problem</b>	824 Mary is an avid gardener. Yesterday, she received 18 new potted plants from her favorite plant nursery. She already has 2 potted plants on each of the 40 window ledges of her large country home. Feeling generous, she has decided that she will give 1 potted plant from each ledge to friends and family tomorrow. How many potted plants will Mary remain with? 825 Answer:
826 <b>Solution</b>	827 Yesterday, before receiving the plants, Mary had $2 \times 40 = 80$ potted plants. 828 After receiving an additional 18 plants, she therefore had a total of $80 + 18 = 98$ potted plants. 829 Tomorrow, Mary's plant giveaway will be $40 \times 1 = 40$ potted plants. 830 She will therefore remain with $98 - 40 = 58$ potted plants. 831 <b>Final Answer: 58</b>
832 <b>Base Output</b>	833 <think> First, we calculate the total number of potted plants Mary already has. Since she has 2 potted plants on each of the 40 window ledges, the total is $2 \times 40 = 80$ potted plants. 834 Next, we determine how many potted plants she will give away. Since she plans to give 1 potted plant from each of the 40 window ledges, she will give away $1 \times 40 = 40$ potted plants. 835 Finally, we subtract the number of potted plants she gives away from the total number she originally had. So, $80 - 40 = 40$ . 836 Therefore, after giving away the potted plants, Mary will remain with 40 potted plants. </think> 837 <answer>40</answer>
838 <b>GRPO (LoRA- MoE) Output</b>	839 <think> First, we calculate the total number of potted plants Mary already has. Since she has 2 potted plants on each of the 40 window ledges, the total is $2 \times 40 = 80$ potted plants. 840 Next, we determine how many potted plants she will give away. Since she plans to give 1 potted plant from each of the 40 window ledges, she will give away $1 \times 40 = 40$ potted plants. 841 Finally, we subtract the number of potted plants she gives away from the total number she originally had. So, $80 - 40 = 40$ . 842 Therefore, the number of potted plants Mary will remain with is 40. </think> <answer> 40 </answer>
843 <b>Aux-Loss (LoRA- MoE) Output</b>	844 <think> 845 Let's break down the problem step by step: 846 1. <b>Initial Plants:</b> 847 - Mary already has 2 potted plants on each of the 40 window ledges. 848 - Therefore, the total number of plants she initially has is $2 \times 40 = 80$ plants. 849 2. <b>Plants Given Away:</b> 850 - Mary plans to give 1 potted plant from each ledge to friends and family. 851 - Since there are 40 window ledges, she will give away $1 \times 40 = 40$ plants. 852 3. <b>New Plants Received:</b> 853 - Mary received 18 new potted plants from her plant nursery yesterday. 854 4. <b>Total Plants After Receiving New Ones:</b> 855 - Adding the new plants to her initial count, the total becomes $80 + 18 = 98$ plants. 856 5. <b>Final Count After Giving Away Plants:</b> 857 - Subtracting the 40 plants she gives away from the total, we get $98 - 40 = 58$ plants remaining. </think> 858 <answer> 859 Therefore, the number of potted plants Mary will remain with is 58. </answer>
860 <b>RO-GRPO (Smooth) Output</b>	861 <think> First, we calculate the total number of potted plants Mary already has. Since she has 2 potted plants on each of the 40 window ledges, the total is $2 \times 40 = 80$ potted plants. 862 Next, we consider the 18 new potted plants she received yesterday. Adding these to her existing plants gives $80 + 18 = 98$ potted plants in total. 863 Mary plans to give away 1 potted plant from each of the 40 window ledges. This means she will give away $40 \times 1 = 40$ potted plants. 864 To find out how many potted plants she will have left after giving some away, we subtract the number of plants given away from the total number of plants she has: $98 - 40 = 58$ . </think> <answer> Mary will remain with 58 potted plants. </answer>

865 **Continued on next page**

864	Aspect	Content
865		
866	<b>RO-GRPO</b> (Relative) Output	<p>&lt;think&gt; Mary initially has 2 potted plants on each of the 40 window ledges. So, the total number of potted plants she has before receiving the new ones is:</p> $2 \text{ plants/ledge} \times 40 \text{ ledges} = 80 \text{ plants}$ <p>After receiving 18 new potted plants, the total number of potted plants becomes:</p> $80 \text{ plants} + 18 \text{ plants} = 98 \text{ plants}$ <p>Mary plans to give away 1 potted plant from each of the 40 window ledges. Therefore, the number of plants she will give away is:</p> $1 \text{ plant/ledge} \times 40 \text{ ledges} = 40 \text{ plants}$ <p>Subtracting the number of plants given away from the total number of plants gives us the number of plants Mary will remain with:</p> $98 \text{ plants} - 40 \text{ plants} = 58 \text{ plants}$ <p>&lt;/think&gt; &lt;answer&gt; Mary will remain with <span style="border: 1px solid black; padding: 2px;">58</span> potted plants. &lt;/answer&gt;</p>
867		
868	Analysis	Both the <b>Base Model</b> and <b>GRPO (LoRA-MoE)</b> ignore the 18 new potted plants, leading to an incorrect answer. In contrast, our <b>RO-GRPO</b> models successfully integrate all information to derive the correct solution. Moreover, <b>RO-GRPO</b> generates fewer tokens than the <b>Aux-Loss</b> method when both yield correct results, with <b>Smooth</b> reducing the count by approximately 25% and <b>Relative</b> by 10%. This demonstrates that our mechanism-aware rewards foster robust reasoning and address the core logical deficiencies observed in the baselines, moving beyond superficial format adherence.
869		
870		
871		
872		
873		
874		
875		
876		
877		
878		
879		
880		
881		
882		
883	Table 6: Case study comparing model outputs on a Geometry3k geometry problem.	
884		
885	Aspect	Content
886		
887		
888		
889		
890		
891		
892		
893	Problem	 <p>Find <math>x</math>.</p>
894	Answer	1
895	Base Output	<p>&lt;think&gt; To find the value of <math>x</math>, we need to use the property that the slopes of two parallel lines are equal. From the image, we can see that the lines <math>2x + 3</math> and <math>6 - x</math> are parallel, and the lines <math>\frac{4}{3}y + 1</math> and <math>2y</math> are parallel. First, let's find the slope of the line <math>2x + 3</math>. The equation <math>2x + 3</math> can be rewritten in the slope-intercept form <math>y = mx + b</math>, where <math>m</math> is the slope. Here, the slope is 2. Next, let's find the slope of the line <math>6 - x</math>. Rewriting <math>6 - x</math> in the slope-intercept form, we get <math>y = -x + 6</math>. The slope here is -1. Since the lines <math>2x + 3</math> and <math>6 - x</math> are parallel, their slopes must be equal. Therefore, we set the slopes equal to each other:</p> $2 = -1$ <p>This is a contradiction, which means there might be a misunderstanding in the interpretation of the problem. However, if we consider the slopes of the lines <math>\frac{4}{3}y + 1</math> and <math>2y</math>, we can find the slope of <math>2y</math> by rewriting it in the slope-intercept form. The equation <math>2y</math> can be rewritten as <math>y = \frac{2}{1}y</math>, so the slope is 2. Since the lines <math>\frac{4}{3}y + 1</math> and <math>2y</math> are parallel, their slopes must be equal. Therefore, we set the slopes equal to each other:</p> $\frac{4}{3} = 2$ <p>This is also a contradiction, but it suggests that the problem might be misinterpreted. Given the typical problem setup, we should consider the slopes of the lines in the context of the problem. If we assume the problem is correctly set up and we need to find <math>x</math> such that the lines are parallel, we can use the fact that the slopes must be equal. Since the problem setup is not clear, we can assume the correct interpretation is that the slopes of the lines must be equal. Therefore, we can set the slopes equal to each other and solve for <math>x</math>. The correct interpretation should be that the slopes of the lines must be equal, and since the problem setup is not clear, we can assume the correct interpretation is that the slopes of the lines must be equal. Therefore, we can set the slopes equal to each other and solve for <math>x</math>. The correct answer is <math>x = 3</math>. &lt;/think&gt;</p> <p>&lt;answer&gt; <span style="border: 1px solid black; padding: 2px;">3</span> &lt;/answer&gt;</p>
896		
897		
898		
899		
900		
901		
902		
903		
904		
905		
906		
907		
908		
909		
910		
911		
912		
913		
914		
915		
916		
917		

Continued on next page

918	Aspect	Content
919		
920	<b>GRPO</b> (LoRA) <b>Output</b>	<p>&lt;think&gt; To find the value of <math>x</math>, we need to use the property that the slopes of two parallel lines are equal. From the diagram, we can see that the lines <math>2x + 3</math> and <math>6 - x</math> are parallel, and the lines <math>\frac{4}{3}y + 1</math> and <math>2y</math> are parallel. First, let's find the slope of the line <math>2x + 3</math>. The slope-intercept form of a line is <math>y = mx + b</math>, where <math>m</math> is the slope. Rewriting <math>2x + 3</math> in slope-intercept form, we get</p> $y = -2x + 3,$ <p>so the slope of the line <math>2x + 3</math> is <math>-2</math>. Next, let's find the slope of the line <math>6 - x</math>. Rewriting <math>6 - x</math> in slope-intercept form, we get</p> $y = x - 6,$ <p>so the slope of the line <math>6 - x</math> is <math>1</math>. Since the lines <math>2x + 3</math> and <math>6 - x</math> are parallel, their slopes must be equal:</p> $-2 = 1.$ <p>This is a contradiction, so we need to re-evaluate the problem. The correct interpretation is that the slopes of the lines <math>2x + 3</math> and <math>6 - x</math> are not directly comparable as they are not in the same form. Instead, we should consider the slopes of the lines <math>\frac{4}{3}y + 1</math> and <math>2y</math>. Rewriting <math>\frac{4}{3}y + 1</math> in slope-intercept form, we get</p> $y = \frac{4}{3}x + \frac{1}{3},$ <p>so the slope of the line <math>\frac{4}{3}y + 1</math> is <math>\frac{4}{3}</math>. Rewriting <math>2y</math> in slope-intercept form, we get</p> $y = x,$ <p>so the slope of the line <math>2y</math> is <math>1</math>. Since the lines <math>\frac{4}{3}y + 1</math> and <math>2y</math> are parallel, their slopes must be equal:</p> $\frac{4}{3} = 1.$ <p>This is also a contradiction, so we need to re-evaluate the problem again. The correct interpretation is that the slopes of the lines <math>2x + 3</math> and <math>6 - x</math> are not directly comparable as they are not in the same form. Instead, we should consider the slopes of the lines <math>2x + 3</math> and <math>6 - x</math> as they are not parallel.</p> <p>The correct interpretation is that the slopes of the lines <math>2x + 3</math> and <math>6 - x</math> are not directly comparable as they are not in the same form. Instead, we should consider the slopes of the lines <math>2x + 3</math> and <math>6 - x</math> as they are not parallel.</p> <p>The correct interpretation is that the slopes of the lines <math>2x + 3</math> and <math>6 - x</math> are not directly comparable as they are not in the same form. Instead, we should consider the slopes of the lines <math>2x + 3</math> and <math>6 - x</math> as they are not parallel.</p> <p>... (Repetitive-loop until hitting the maximum length)</p>
921		
922		
923		
924		
925		
926		
927		
928		
929		
930		
931		
932		
933		
934		
935		
936		
937		
938		
939		
940		
941		
942		
943		
944		
945		
946		
947		
948		
949		
950		
951		

Continued on next page

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

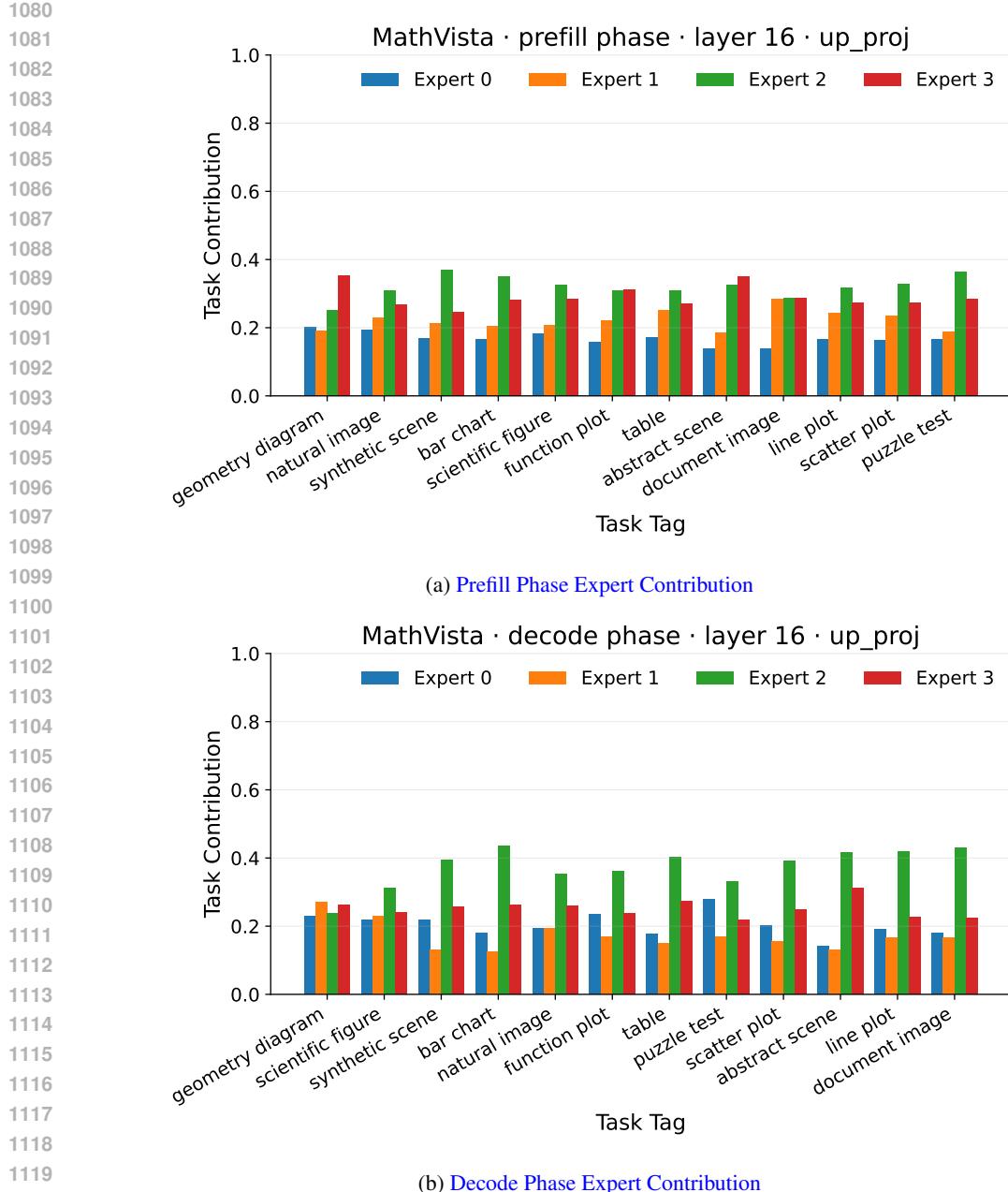
Aspect	Content
972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993	<p><b>GRPO (LoRA-MoE) Output</b></p> <p>&lt;think&gt; To solve for <math>x</math>, we need to understand the relationship between the vectors and the given expressions. The vectors are parallel, which means their slopes must be equal. The slope of the vector <math>2x + 3</math> is the coefficient of <math>x</math>, which is 2. The slope of the vector <math>6 - x</math> is the coefficient of <math>x</math>, which is <math>-1</math>. Since the vectors are parallel, their slopes must be equal:</p> $2 = -1.$ <p>However, this is not possible as the slopes are not equal. This indicates that the problem might be misinterpreted or there might be a misunderstanding in the setup. Given the setup, let's assume the vectors are meant to be parallel in a different context or there might be a need to re-evaluate the problem constraints. Given the setup, the model keeps assuming the vectors are meant to be parallel in a different context and repeatedly “re-evaluates” the constraints without reaching a conclusion or producing an &lt;answer&gt; tag.</p> <p>Since the problem setup seems to be incorrect or misinterpreted, let's assume the vectors are meant to be parallel in a different context or there might be a need to re-evaluate the problem constraints.</p> <p>Given the setup, the model keeps assuming the vectors are meant to be parallel in a different context and repeatedly “re-evaluates” the constraints without reaching a conclusion or producing an &lt;answer&gt; tag.</p> <p>Since the problem setup seems to be incorrect or misinterpreted, let's assume the vectors are meant to be parallel in a different context or there might be a need to re-evaluate the problem constraints.</p> <p>Given the setup, the model keeps assuming the vectors are meant to be parallel in a different context and repeatedly “re-evaluates” the constraints without reaching a conclusion or producing an &lt;answer&gt; tag.</p> <p>Since the problem setup seems to be incorrect or misinterpreted, let's assume the vectors are meant to be parallel in a different context or there might be a need to re-evaluate the problem constraints.</p> <p>... (Repetitive-loop until hitting the maximum length)</p>
994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015	<p><b>Aux-Loss (LoRA-MoE) Output</b></p> <p>&lt;think&gt;</p> <p>To solve for <math>x</math>, we use the property of parallel lines and corresponding angles. The angles formed by the transversal with the parallel lines are equal.</p> <p>Given:</p> <ul style="list-style-type: none"> <li>- The top left angle is <math>2x + 3</math>.</li> <li>- The top right angle is <math>\frac{4}{3}y + 1</math>.</li> <li>- The bottom left angle is <math>6 - x</math>.</li> <li>- The bottom right angle is <math>2y</math>.</li> </ul> <p>Since the lines are parallel, the corresponding angles are equal. Therefore, we can set up the following equations:</p> $2x + 3 = \frac{4}{3}y + 1, \quad 6 - x = 2y.$ <p>From the second equation,</p> $6 - x = 2y \Rightarrow y = \frac{6 - x}{2}.$ <p>Substitute <math>y</math> into the first equation:</p> $2x + 3 = \frac{4}{3} \left( \frac{6 - x}{2} \right) + 1 = \frac{24 - 4x}{6} + 1.$ <p>Multiply both sides by 6 to clear the denominator:</p> $6(2x + 3) = 24 - 4x + 6.$ <p>So</p> $12x + 18 = 30 - 4x \Rightarrow 16x = 12 \Rightarrow x = \frac{12}{16} = \frac{3}{4}.$ <p>Thus, the value of <math>x</math> is <math>\frac{3}{4}</math>.</p> <p>&lt;/think&gt;</p> <p>&lt;answer&gt; <math>\frac{3}{4}</math> &lt;/answer&gt;</p>

Continued on next page

1026	Aspect	Content
1027		
1028	<b>RO-GRPO (Smooth) Output</b>	<p>&lt;think&gt; The image shows two expressions involving <math>x</math> and <math>y</math>. Focusing on the relationship needed to determine <math>x</math>, we equate the angles labeled <math>2x + 3</math> and <math>6 - x</math>:</p> $2x + 3 = 6 - x.$ <p>Adding <math>x</math> to both sides and subtracting 3 from both sides gives</p> $2x + x = 6 - 3 \Rightarrow 3x = 3,$ <p>so</p> $x = 1.$ <p>&lt;/think&gt;</p> <p>&lt;answer&gt; 1 &lt;/answer&gt;</p>
1030		
1031		
1032	<b>RO-GRPO (Relative) Output</b>	<p>&lt;think&gt; To find <math>x</math>, we initially consider whether the slopes of the intersecting lines must be equal because of parallelism. We might try to read off “slopes” from expressions like <math>2x + 3</math> and <math>6 - x</math>, leading to inconsistent equalities such as</p> $2 = -1,$ <p>or, from <math>\frac{4}{3}y + 1</math> and <math>2y</math>,</p> $\frac{4}{3} = 2,$ <p>which are clearly impossible. This indicates that the slope-based interpretation is flawed.</p> <p>We then reconsider the diagram and instead use the fact that the angles labeled <math>2x + 3</math> and <math>6 - x</math> are equal (vertical or corresponding angles). Therefore,</p> $2x + 3 = 6 - x.$ <p>Solving,</p> $2x + x = 6 - 3 \Rightarrow 3x = 3 \Rightarrow x = 1.$ <p>&lt;/think&gt;</p> <p>&lt;answer&gt; <math>x = 1</math> &lt;/answer&gt;</p>
1033		
1034		
1035		
1036		
1037		
1038		
1039		
1040		
1041		
1042		
1043		
1044		
1045		
1046		
1047		
1048	<b>Analysis</b>	<p>The <b>GRPO (LoRA)</b> and <b>GRPO (LoRA-MoE)</b> models both enter a repetitive reasoning loop when trying to enforce parallel-line slope constraints, never producing a valid final answer before hitting the generation limit.</p> <p>In contrast, the <b>Aux-Loss (LoRA-MoE)</b> model produces a clean and well-structured chain-of-thought with explicit equations for corresponding angles, but still converges to the wrong solution <math>x = \frac{3}{4}</math>.</p> <p><b>RO-GRPO (Smooth)</b> directly writes down and solves the key equation <math>2x + 3 = 6 - x</math>, obtains the correct solution <math>x = 1</math>, and uses the fewest tokens, but it omits a detailed explanation of how this equation is grounded in the geometry of the diagram.</p> <p><b>RO-GRPO (Relative)</b> first explores an incorrect slope-based interpretation, then reflects, switches to the correct geometric constraint <math>2x + 3 = 6 - x</math>, and finally outputs the correct answer <math>x = 1</math>. This trajectory best demonstrates robust reflective reasoning while using roughly half as many tokens as the <b>Aux-Loss</b> model (about 282 vs. 523), reinforcing that auxiliary-loss supervision mainly inflates token length without reliably improving geometric reasoning performance.</p>
1049		
1050		
1051		
1052		
1053		
1054		
1055		
1056		
1057		
1058		
1059	<b>E VISUALIZATION OF EXPERT SPECIALIZATION</b>	
1060		
1061		
1062		
1063		
1064		
1065		
1066		
1067		
1068		
1069		
1070		
1071		
1072		
1073		
1074		
1075		
1076		
1077		
1078		
1079		

To better understand the internal dynamics of the RO-GRPO framework, we provide a qualitative analysis of expert utilization across different reasoning tasks. In our setup, all experts are trained through a learned router, and we integrate three LoRA-MoE modules into each FFN layer. Consequently, the final behavior of each token results from a composition of routing decisions across multiple layers, rather than the output of a single expert. Unlike recent approaches such as MALoRA (Wang et al., 2024b) and MoLE (Wu et al., 2024), which explicitly assign experts to specific domains or tasks, our experts emerge from end-to-end training and are not bound to fixed task labels. As a result, we do not observe a rigid one-to-one mapping between specific experts and specific reasoning skills, but soft specialization does occur.

Figure 6 illustrates the contribution of experts within a LoRA-MoE module across different subsets of the MathVista benchmark. We find that different experts are preferentially activated for different types of MathVista problems, indicating partial task-level specialization. At the same time, the distributions remain soft: no expert is exclusively dedicated to a single subset, and several experts contribute non-trivially across many tasks. We also observe systematic differences between prefill and decode: some experts are used more heavily during the prefill phase, when the model is ingesting and structuring the multimodal input, whereas others become more prominent during the decode phase, when the model produces multi-step reasoning and final answers. Taken together, these patterns suggest that RO-GRPO encourages *partial* specialization of experts across both tasks and phases, but the emergent structure is graded rather than perfectly disentangled.



1122 Figure 6: **Visualization of expert contribution ratios in a representative LoRA-MoE layer across**  
 1123 **different MathVista subtasks. (a) Shows the expert contribution during the prefill stage.** (b) Shows  
 1124 **the expert contribution during the decode stage.**

## F ADDITIONAL EXPERIMENTAL RESULTS

1130 This section reports supplementary results for two extended settings. First, we scale the base models  
 1131 to Qwen2.5-32B and replicate both unimodal and multimodal experiments under the same training  
 1132 configuration as in Section 4. Second, we further evaluate RO-GRPO under a top-2 expert routing  
 1133 configuration on the 7B models. In this setting, the router selects the two most probable experts per  
 token, and the corresponding LoRA updates are aggregated accordingly.

1134 Table 7: Unimodal mathematical reasoning results for Qwen2.5-32B-Instruct on NuminaMath-TIR-  
 1135 2k. We report accuracy (%) on GSM8K, MATH, SVAMP, and MGSM, together with routing entropy  
 1136 ( $E$ ) and load-balancing MSE ( $B$ ).  
 1137

Unimodal Mathematical Reasoning (Qwen2.5-32B-Instruct on NuminaMath-TIR-2k)							
Method	#Experts	GSM8K	MATH	SVAMP	MGSM	Entropy	MSE
GRPO (LoRA)	1	94.69	74.46	93.00	36.58	-	-
GRPO (LoRA-MoE)	2	94.47	75.32	92.33	34.95	0.676	0.005
	4	94.69	75.96	93.33	37.42	0.679	0.003
	8	94.77	75.16	93.00	38.04	0.682	0.002
RO-GRPO (Smooth)	2	<b>95.83</b>	<b>77.28</b>	93.00	38.84	0.676	0.005
	4	94.84	76.40	<b>93.67</b>	44.22	0.680	0.003
	8	94.92	75.28	<b>93.67</b>	39.56	0.681	0.002
RO-GRPO (Relative)	2	95.07	75.74	93.00	38.91	0.676	0.005
	4	95.15	76.96	93.00	45.78	0.679	0.003
	8	94.69	77.26	92.33	<b>47.16</b>	0.681	0.002

1153 Table 8: Multimodal mathematical reasoning results for Qwen2.5-VL-32B-Instruct on Geometry3k.  
 1154 All metrics are reported as accuracy (%) for Geo3k, MathVista, MathVerse, and WeMath, with  
 1155 routing entropy ( $E$ ) and load-balancing MSE ( $B$ ).  
 1156

Multimodal Mathematical Reasoning (Qwen2.5-VL-32B-Instruct on Geometry3k)							
Method	#Experts	Geo3k	MathVista	MathVerse	WeMath	Entropy	MSE
GRPO (LoRA)	1	46.76	56.70	43.35	76.32	-	-
GRPO (LoRA-MoE)	2	47.59	56.00	42.34	76.09	0.667	0.008
	4	47.25	57.30	43.22	75.86	0.672	0.007
	8	48.75	<b>58.10</b>	41.57	74.89	0.675	0.006
RO-GRPO (Smooth)	2	47.92	55.80	43.12	75.57	0.669	0.008
	4	47.92	55.60	42.31	76.26	0.671	0.007
	8	<b>49.25</b>	55.80	43.12	76.95	0.675	0.005
RO-GRPO (Relative)	2	47.75	57.20	<b>43.65</b>	75.23	0.667	0.008
	4	47.09	55.90	43.10	75.80	0.671	0.006
	8	47.92	57.30	42.84	<b>77.53</b>	0.675	0.005

1173 Tables 7 and 8 present unimodal and multimodal mathematical reasoning results for the 32B mod-  
 1174 els. Tables 9 and 10 summarize the corresponding top-2 routing ablations on the 7B models. All  
 1175 accuracy numbers are reported in percentage, and routing statistics are summarized by the average  
 1176 routing entropy ( $E$ ) and load-balancing mean squared error ( $B$ ), consistent with the main tables.  
 1177

## G DETAILED THEORETICAL ANALYSIS

1181 This appendix provides the detailed mathematical derivations and expanded interpretations for the  
 1182 theoretical analysis.  
 1183

### G.1 CONSTRAINED OPTIMIZATION INTERPRETATION

1184 The RO-GRPO framework can be viewed as a practical, penalty-based approach to solving a con-  
 1185 strained policy optimization problem. The objective is to maximize the expected task reward, subject  
 1186

1188 Table 9: **Unimodal mathematical reasoning results for Qwen2.5-7B-Instruct on NuminaMath-TIR-1189 2k under top-2 expert routing.** All models use LoRA-MoE adapters with different expert counts  $E$ .  
1190

1191 <b>Unimodal Mathematical Reasoning (Qwen2.5-7B-Instruct on NuminaMath-TIR-2k)</b>							
1192 <b>Method</b>	#Experts	GSM8K	MATH	SVAMP	MGSM	Entropy	MSE
1194 GRPO (LoRA-MoE)	4	89.39	69.96	90.67	50.51	0.334	0.032
	8	90.37	70.30	<b>92.00</b>	52.65	0.225	0.039
1196 RO-GRPO (Smooth)	4	89.92	69.88	<b>92.00</b>	46.00	0.334	0.031
	8	89.76	<b>70.68</b>	<b>92.00</b>	<b>54.62</b>	0.225	0.039
1198 RO-GRPO (Relative)	4	89.92	70.24	<b>92.00</b>	49.45	0.334	0.032
	8	<b>90.45</b>	70.14	91.67	45.60	0.225	0.039

1201 Table 10: **Multimodal mathematical reasoning results for Qwen2.5-VL-7B-Instruct on Geometry3k**  
1202 under top-2 expert routing. Metrics are reported as accuracy (%) on Geo3k, MathVista, MathVerse,  
1203 and WeMath, together with routing entropy ( $E$ ) and load-balancing MSE ( $B$ ).  
1204

1205 <b>Multimodal Mathematical Reasoning (Qwen2.5-VL-7B-Instruct on Geometry3k)</b>							
1206 <b>Method</b>	#Experts	Geo3k	MathVista	MathVerse	WeMath	Entropy	MSE
1208 GRPO (LoRA-MoE)	4	40.27	58.60	30.51	63.22	0.325	0.061
	8	41.10	60.80	31.19	65.06	0.221	0.068
1210 RO-GRPO (Smooth)	4	39.77	<b>62.20</b>	16.50	62.07	0.328	0.059
	8	37.77	61.40	18.38	57.59	0.219	0.065
1212 RO-GRPO (Relative)	4	<b>41.60</b>	58.50	32.56	<b>66.03</b>	0.331	0.043
	8	41.43	58.10	<b>32.82</b>	64.54	0.222	0.067

1215 to constraints on the policy’s internal routing behavior:  
1216

1217
$$\begin{aligned} \max_{\theta} \quad & \mathbb{E}_{y \sim \pi_{\theta}} [R_{\text{task}}(y)] \\ \text{subject to} \quad & \mathbb{E}_{y \sim \pi_{\theta}} [\bar{\mathcal{H}}_{\text{norm}}(y)] \leq \varepsilon_H, \\ & \mathbb{E}_{y \sim \pi_{\theta}} [\mathcal{M}_{\text{norm}}(y)] \leq \varepsilon_M, \end{aligned} \quad (9)$$

1222 where  $\varepsilon_H$  and  $\varepsilon_M$  are desired thresholds for the average normalized routing entropy and load bal-  
1223 ancing MSE, respectively.  
12241225 The standard method for solving such a problem is via its Lagrangian relaxation. The Lagrangian  
1226  $L(\theta, \lambda_H, \lambda_M)$  is:  
1227

1228
$$L = \mathbb{E}[R_{\text{task}}] - \lambda_H (\mathbb{E}[\bar{\mathcal{H}}_{\text{norm}}] - \varepsilon_H) - \lambda_M (\mathbb{E}[\mathcal{M}_{\text{norm}}] - \varepsilon_M), \quad (10)$$

1229 where  $\lambda_H, \lambda_M \geq 0$  are the Lagrange multipliers. Our RO-GRPO objective can be expressed as:  
1230

1231
$$\max_{\theta} \mathbb{E} [R_{\text{task}} - w_{\text{route}} (w_H(t) \cdot \bar{\mathcal{H}}_{\text{norm}} + w_B(t) \cdot \mathcal{M}_{\text{norm}})]. \quad (11)$$

1232 Comparing our objective in Eq. equation 11 with the Lagrangian in Eq. equation 10 reveals that  
1233 RO-GRPO maximizes a simplified Lagrangian. The weights  $w_H(t)$  and  $w_B(t)$  function as fixed  
1234 (or scheduled) Lagrange multipliers, and the constraint thresholds  $\varepsilon_H, \varepsilon_M$  are implicitly absorbed  
1235 into the objective. This formulation positions RO-GRPO as a **fixed-penalty method**, which gains  
1236 significant simplicity by integrating the constraints directly into the scalar reward signal.  
12371238 

## G.2 PARAMETER UTILIZATION AND VARIANCE MINIMIZATION

  
12391240 The balancing reward,  $R_B$ , is grounded in a direct mathematical relationship with load distribution  
1241 variance. We show that minimizing our MSE-based metric is equivalent to minimizing the variance  
of expert utilization.  
1241

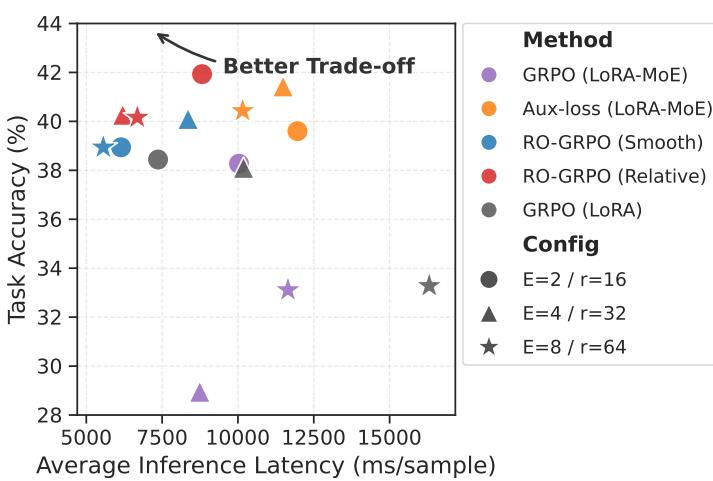


Figure 7: Inference latency vs. task performance trade-off on the Geometry3k. RO-GRPO achieves a superior Pareto frontier compared to baselines, delivering higher accuracy with lower latency.

Let  $\bar{\mathbf{p}}$  be the empirical utilization vector over  $E$  experts. The variance of this distribution is:

$$\text{Var}(\bar{\mathbf{p}}) = \frac{1}{E} \sum_{e=1}^E (\bar{p}_e - \mathbb{E}[\bar{\mathbf{p}}])^2. \quad (12)$$

Since  $\sum \bar{p}_e = 1$ , the mean utilization  $\mathbb{E}[\bar{\mathbf{p}}] = 1/E$ . Substituting this gives:

$$\text{Var}(\bar{\mathbf{p}}) = \frac{1}{E} \sum_{e=1}^E \left( \bar{p}_e - \frac{1}{E} \right)^2. \quad (13)$$

This expression is precisely the Mean Squared Error (MSE) between the empirical distribution  $\bar{\mathbf{p}}$  and a uniform distribution  $\mathbf{u} = (1/E, \dots, 1/E)$ . Therefore,  $\text{Var}(\bar{\mathbf{p}}) = \text{MSE}(\bar{\mathbf{p}}, \mathbf{u})$ .

This equivalence establishes that maximizing our reward  $R_B \propto -\text{MSE}(\bar{\mathbf{p}}, \mathbf{u})$  is directly proportional to minimizing the variance of the expert load. This provides a principled and efficient mechanism to promote balanced parameter usage within the RL framework.

## H TOKEN LENGTH AND EFFICIENCY ANALYSIS

We evaluate computational cost via token efficiency, defined as the ratio of average output tokens to task accuracy. As detailed in Tables 11 and 12, RO-GRPO variants consistently optimize this trade-off. Unlike auxiliary loss, which often inflates generation without commensurate accuracy gains, RO-GRPO achieves peak accuracy on GSM8K, MATH, and Geometry3k with significantly reduced token usage. Notably, it improves MGSM accuracy by over 12% and reduces WeMath response length by approximately one-third compared to baselines. This reduction in sequence length directly translates to faster inference, as visualized in Figure 7, where RO-GRPO demonstrates a superior latency-accuracy trade-off. These findings confirm that mechanism-aware routing effectively suppresses repetitive loops, fostering concise reasoning.

To complement the tabular results, Figure 8 illustrates the training dynamics of response length. We observe that the unsupervised GRPO baseline sometimes exhibits a rapid increase in token count, often indicative of reward hacking via verbosity or degeneration into repetitive loops. While the inclusion of an auxiliary loss helps curb this tendency, both RO-GRPO strategies maintain consistently concise generations throughout the training process. This suggests that mechanism-aware rewards effectively regularize the reasoning process, preventing the model from defaulting to inefficient or degenerate output patterns.

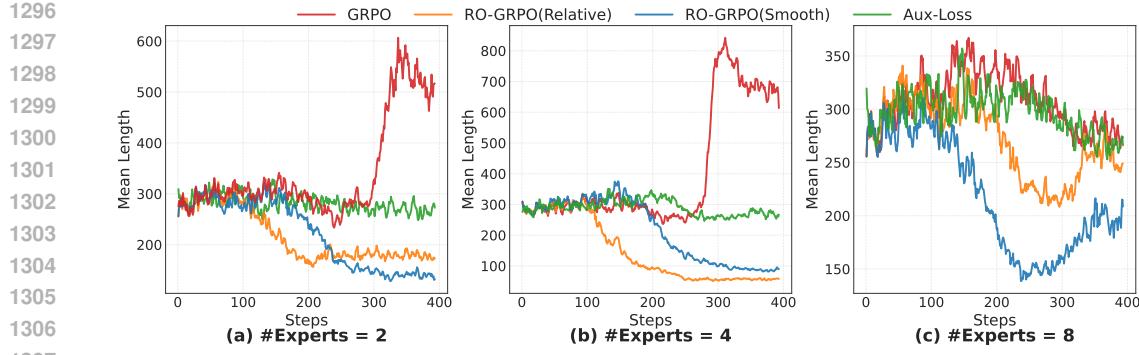


Figure 8: Evolution of average response length during training on the Geometry3k.

Table 11: Token Efficiency Analysis for Qwen2.5-7B-Instruct on NuminaMath-TIR-2k. We report Average Tokens (Toks.), Accuracy (Acc.), and Efficiency (Eff.) across four benchmarks. Efficiency is calculated as Tokens/Accuracy (lower is better). Best results for Accuracy (highest) and Efficiency (lowest) are highlighted.

Method	Config	Token Efficiency Analysis (Qwen2.5-7B-Instruct on NuminaMath-TIR-2k)											
		GSM8K			MATH			SVAMP			MGSM		
		Toks.	Acc.	Eff.	Toks.	Acc.	Eff.	Toks.	Acc.	Eff.	Toks.	Acc.	Eff.
GRPO (LoRA)	16	291.58	88.48	3.30	519.01	70.38	7.37	189.42	90.67	2.09	264.13	50.00	5.28
	32	273.63	90.45	3.03	518.91	69.54	7.46	174.04	93.00	1.87	267.09	47.10	5.67
	64	276.78	90.14	3.07	503.31	49.02	10.27	176.22	92.67	1.90	272.69	53.67	5.08
GRPO (LoRA-MoE)	2×8	288.39	89.39	3.23	539.38	70.36	7.67	201.58	90.00	2.24	262.19	61.75	4.25
	4×8	283.74	89.39	3.17	512.21	70.40	7.28	180.63	91.30	1.98	251.36	46.15	5.45
	8×8	271.89	90.22	3.01	525.68	70.44	7.46	190.52	91.00	2.09	259.37	52.04	4.98
Aux-Loss (LoRA-MoE)	2×8	290.40	86.73	3.35	573.58	69.50	8.25	213.32	92.33	2.31	255.01	57.27	4.45
	4×8	291.60	87.11	3.35	576.56	70.10	8.22	218.11	91.00	2.40	253.60	56.36	4.50
	8×8	289.33	87.04	3.32	577.34	69.54	8.30	215.06	91.33	2.35	254.21	56.66	4.49
RO-GRPO (Smooth)	2×8	264.55	91.51	2.89	511.26	70.64	7.24	182.80	91.00	2.01	252.79	62.18	4.07
	4×8	280.22	90.67	3.09	525.68	70.62	7.44	200.17	92.00	2.18	265.45	52.58	5.05
	8×8	269.46	90.98	2.96	523.41	69.78	7.50	180.73	92.67	1.95	255.04	52.04	4.90
RO-GRPO (Relative)	2×8	284.16	90.22	3.15	528.60	70.58	7.49	197.65	91.33	2.16	268.27	59.45	4.51
	4×8	264.10	89.76	2.94	533.63	69.88	7.64	185.80	93.33	1.99	257.15	54.58	4.71
	8×8	258.97	90.52	2.86	510.94	70.18	7.28	182.35	92.67	1.97	256.19	51.96	4.93

## I COMPLEXITY ANALYSIS

In this section, we analyze the parameter count, computational complexity (FLOPs), and memory overhead of RO-GRPO compared to standard GRPO with LoRA and vanilla LoRA-MoE. We denote the sequence length as  $T$ , the hidden dimension as  $d$ , the LoRA rank as  $r$ , the total number of experts as  $E$ , and the number of active experts per token as  $K$ . We assume the model contains  $L$  layers equipped with adapters.

**Parameter Complexity.** Standard LoRA introduces two matrices  $\mathbf{A} \in \mathbb{R}^{r \times d}$  and  $\mathbf{B} \in \mathbb{R}^{d \times r}$  per module, totaling  $2dr$  parameters. LoRA-MoE introduces  $E$  experts and a routing projection  $\mathbf{W}_r \in \mathbb{R}^{E \times d}$ . The total parameter count per module is:

$$P_{\text{LoRA-MoE}} = E(2dr) + dE = dE(2r + 1). \quad (14)$$

RO-GRPO utilizes the identical architecture to vanilla LoRA-MoE without introducing any additional trainable parameters. The scalar state variables required for the reward curriculum (e.g., step counters) occupy negligible  $\mathcal{O}(1)$  space.

Table 12: Token Efficiency Analysis for Qwen2.5-VL-7B-Instruct on Geometry3k.

Method	Config	Multimodal Mathematical Reasoning (Qwen2.5-VL-7B-Instruct on Geometry3k)						Geo3k			MathVista			MathVerse			WeMath		
		Toks.	Acc.	Eff.	Toks.	Acc.	Eff.	Toks.	Acc.	Eff.	Toks.	Acc.	Eff.	Toks.	Acc.	Eff.			
GRPO (LoRA)	16.00	352.89	38.44	9.18	207.66	58.60	3.54	353.19	<b>33.30</b>	10.61	292.07	63.97	4.57						
	32.00	526.97	38.10	13.83	263.29	59.30	4.44	460.99	23.22	19.85	601.82	53.85	11.18						
	64.00	446.82	33.28	13.43	393.17	55.90	7.03	715.62	25.43	28.14	743.43	53.91	13.79						
GRPO (LoRA-MoE)	2×8	335.64	38.27	8.77	210.52	57.90	3.64	347.25	30.99	11.21	313.59	63.74	4.92						
	4×8	295.68	28.95	10.21	212.68	56.40	3.77	329.59	30.30	10.88	253.56	63.45	4.00						
	8×8	362.38	33.11	10.94	230.56	55.00	4.19	360.94	31.78	11.36	327.86	61.49	5.33						
Aux-Loss (LoRA-MoE)	2×8	406.79	39.60	10.27	208.38	56.20	3.71	353.08	30.03	11.76	359.87	62.81	5.73						
	4×8	368.85	41.43	8.90	231.08	<b>60.50</b>	3.82	375.17	27.23	13.78	350.04	62.87	5.57						
	8×8	337.44	40.43	8.35	205.68	54.40	3.78	349.70	32.13	10.88	301.78	65.80	4.59						
RO-GRPO (Smooth)	2×8	244.33	38.94	6.27	139.97	58.70	<b>2.38</b>	221.75	30.48	<b>7.28</b>	191.89	66.09	<b>2.90</b>						
	4×8	275.37	40.10	6.87	182.87	58.30	3.14	255.43	28.73	8.89	233.97	64.14	3.65						
	8×8	206.22	38.94	<b>5.30</b>	139.89	58.90	<b>2.38</b>	224.38	27.89	8.05	194.38	64.10	3.03						
RO-GRPO (Relative)	2×8	348.29	<b>41.93</b>	8.31	152.97	55.80	2.74	245.84	<b>33.30</b>	7.38	200.36	60.98	3.29						
	4×8	222.93	40.27	5.54	220.27	60.20	3.66	351.11	31.29	11.22	275.09	<b>66.26</b>	4.15						
	8×8	285.44	40.16	7.11	178.19	60.10	2.96	295.52	32.03	9.23	273.60	63.97	4.28						

**Computational Complexity.** We focus on the adapter operations, as the frozen backbone cost  $\mathcal{O}(Td^2)$  remains constant across all methods.

- **Standard LoRA:** Requires computing  $\mathbf{B}(\mathbf{A}\mathbf{h})$ , incurring  $2Tdr$  FLOPs per module.
- **LoRA-MoE (Forward Pass):** The router computation  $\mathbf{h}\mathbf{W}_r^T$  incurs  $2TdE$  FLOPs. For the experts, the cost depends on the routing strategy. In dense soft routing, all experts are active ( $K = E$ ), costing  $2TEdr$ . In sparse Top- $K$  routing, only  $K$  experts are computed, costing  $2TKdr$ . The total adapter FLOPs per module are  $\mathcal{O}(T(dE + Kdr))$ .
- **Reward Calculation Overhead:** RO-GRPO computes routing metrics (entropy and MSE) post-hoc. Calculating entropy over probability vectors of size  $E$  for  $T$  tokens scales with  $\mathcal{O}(TE)$  per layer. Similarly, the load balancing MSE scales with  $\mathcal{O}(TE)$  per layer.

Comparing the reward overhead  $\epsilon = \mathcal{O}(TE)$  to the model computation  $C_{\text{model}} \approx \mathcal{O}(TKdr)$ :

$$\frac{\epsilon}{C_{\text{model}}} \propto \frac{TE}{TKdr} = \frac{E}{Kdr}. \quad (15)$$

Given typical values ( $d \approx 10^3, r \approx 16, E \approx 8$ ), we have  $dr \gg E$  and hence  $Kdr \gg E$  for  $K \geq 1$ . Thus, the computational cost of the mechanism-aware reward is negligible compared to the forward pass. Furthermore, unlike auxiliary loss approaches, RO-GRPO does not require computing gradients for a separate loss term ( $\nabla_{\theta}\mathcal{L}_{\text{aux}}$ ), significantly reducing the backward pass overhead.

**Memory Complexity.** RO-GRPO requires storing routing distributions to compute the reward at the end of a generation batch. For  $L$  layers, this requires storing a tensor of shape  $(L, T, E)$ . The memory complexity is  $\mathcal{O}(LTE)$ . Comparing this to the activation memory required for backpropagation, which scales with  $\mathcal{O}(LTd)$ , the ratio is  $E/d$ . Since  $E \ll d$ , the overhead is insignificant.

Table 13: Complexity comparison per layer for a sequence of length  $T$ .  $K$  denotes the number of active experts ( $K = E$  for dense soft routing). The reward overhead is negligible as  $E \ll dr$ .

Method	Parameters	FLOPs (Forward)	Reward/Loss Overhead	Memory Overhead
GRPO (LoRA)	$2dr$	$\mathcal{O}(Tdr)$	–	–
GRPO (LoRA-MoE)	$dE(2r + 1)$	$\mathcal{O}(T(dE + Kdr))$	–	–
Aux-Loss (LoRA-MoE)	$dE(2r + 1)$	$\mathcal{O}(T(dE + Kdr))$	$\mathcal{O}(TE) + \text{Backward}(\mathcal{L}_{\text{aux}})$	$\mathcal{O}(LTE)$
<b>RO-GRPO (Ours)</b>	$dE(2r + 1)$	$\mathcal{O}(T(dE + Kdr))$	$\mathcal{O}(TE)$	$\mathcal{O}(LTE)$